инженерия программного обеспечения

Компьютерные инструменты в образовании, 2018
№ 3: 38–48
УДК: 004.272 : 004.051 + 004.021
http://ipo.spb.ru/journal
doi:10.32603/2071-2340-3-38-48

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ОПТИМИЗАЦИИ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА С УЧЁТОМ И БЕЗ УЧЁТА ВРЕМЕНИ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ

Аль-Марди М. Х. А.

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), Санкт-Петербург, Россия

Аннотация

В данной статье предлагается анализ разработанных нами методов оптимизации параллельного алгоритма с учётом и без учёта времени выполнения каждой операции. Данные методы могут быть применены как к последовательным алгоритмам для получения их параллельного аналога, так и к параллельным алгоритмам с целью повышения их качества. Предлагаемые методы оптимизации параллельного алгоритма позволяют уменьшить объём коммуникаций между процессорами и соответственно сократить время выполнения всего алгоритма.

Ключевые слова: *оптимизация, алгоритм, информационный граф, список следования, время выполнения, операция, процесс, информационная зависимость, единица времени.*

Цитирование: Аль-Марди М. Х. А. Сравнительный анализ методов оптимизации параллельного алгоритма с учётом и без учёта времени выполнения операций // Компьютерные инструменты в образовании. 2018. № 3. С. 38–48.

1. ВВЕДЕНИЕ

Имеется большое количество важнейших задач, решение которых требует использования огромных вычислительных мощностей, зачастую недоступных для современных вычислительных систем. Одним из показателей качества параллельной программы является плотность загрузки вычислительных узлов. Временные задержки при передаче данных по каналам связи от одного процессора к другому приводят к суммарно длительным простоям процессоров и увеличению в целом времени работы алгоритма [1, 2].

Цель планирования информационного графа заключается в том, чтобы минимизировать общее время завершения программы за счет надлежащего распределения задач процессорам и организации последовательности выполнения задач [3]. Планирование выполняется таким образом, что сохраняются ограничения приоритета среди программных задач [4]. Общее время выполнения параллельной программы обычно называется длиной расписания или makespan [5, 6].

В данной статье предлагается анализ методов построения эффективного алгоритма по числу использованных процессоров, времени выполнения алгоритма и объему межпроцессорных передач. Данные методы могут быть применены как к последовательным

алгоритмам для получения их параллельного аналога, так и к параллельным алгоритмам с целью повышения их качества. Также приведен анализ результатов первого и второго этапа решения задач получения расписания выполнения алгоритма, соответствующего заданному информационному графу [7, 8].

2. МЕТОД ОПТИМИЗАЦИИ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ПО ОБЪЁМУ МЕЖПРОЦЕССОРНЫХ КОММУНИКАЦИЙ БЕЗ УЧЁТА ВРЕМЕНИ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ

Метод заключается в следующем:

1. Положить за основу группы вершин, соответствующих ярусам, полученные произвольным способом. Лучше, если эти группы будут получены формализованным способом, например методом оптимизации информационного графа по ширине с помощью матрицы или списка смежности.
2. Процесс перестановки вершин начинается с последней группы. Допустим всего групп m, тогда номер очередной группы $k = m$.
   В первую очередь необходимо в расписание поставить (с учётом групп) вершины, у которых бинарная связь. И только потом расставлять вершины с множественными связями, так как в этом случае существование пузыря неизбежно.
3. В k-й группе выбрать первую вершину. Считать номер позиции этой вершины в группе равным 1: $i = 1$.
4. Сравнить вершину M_{k_i} (где i — номер позиции вершины в группе) с вершинами предыдущей $(k-1)$-й группы. Если в $(k-1)$-й группе существует вершина (M_{k-1_j}, где j — номер позиции вершины в группе, $j \geqslant i$) напрямую связанная в информационном графе ребром с данной вершиной, то вершину M_{k-1_j}, необходимо переместить в своей группе в i-ю позицию. Если в $(k-1)$-й группе нет вершины, связанной с вершиной M_{k_i}, то шаг 6.
5. Если $k > 2$, то $k = k - 1$ и шаг 4.
6. Если в m-й группе перебраны еще не все вершины, то $k = m$, $i = i + 1$ и шаг 4.
7. Если в последней группе перебраны все вершины и $m > 2$, то $m = m - 1$ и шаг 6.
8. Если $m = 1$, то конец метода.

Рассмотрим пример 1. Пусть задан информационный граф алгоритма (рис. 1).

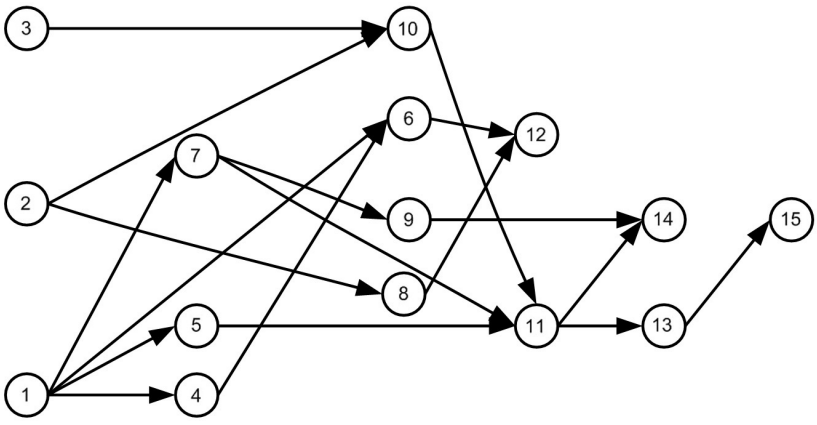


Рис. 1. Информационный граф алгоритма с матрицей смежности

Получим следующие соответствующие ярусам начальные группы вершин графа (рис. 2):

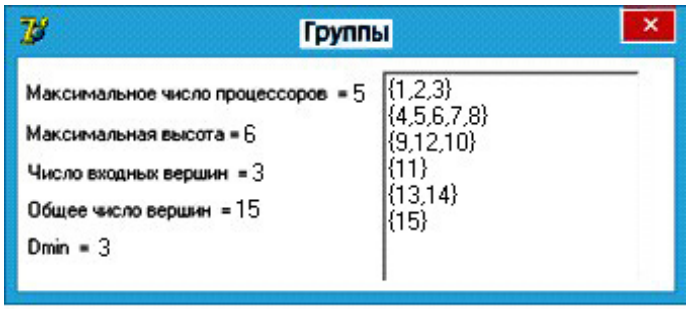


Рис. 2. Начальное расписание алгоритма

Начальное расписание алгоритма (начальные группы) показано на рис. 1. Построим расписание выполнения алгоритма по полученным группам на вычислительной системе с учетом межпроцессорных передач данных. До оптимизации по ширине расписание будет выглядеть как на (рис. 3 а), где P_i — номер процессора, $i = \overline{1,5}$; символ подчеркивания ‘ _ ’ — простой процессора («пузырь») в ожидании получения входных данных от других операций.

P_1: 1, 4, _, _, 9, _, 11, 13, 15;

P_2: 2, _, 5, _, 12, _, _, _, 14;

P_3: 3, _, 6, _, 10;

P_4: _, _, 7;

P_5: _, _, 8.

В соответствии с этим расписанием общее время работы алгоритма $t = 9$, число процессоров$n = 5$, суммарный объем простоев $p = 14$. После оптимизации по ширине получим следующие группы графа:

{1, 2, 3}
{4, 5, 6, 7}
{9, 10, 8} (1)
{11, 12}
{13, 14}
{15}

В соответствии с полученными группами построим временную диаграмму алгоритма (рис. 3 б).

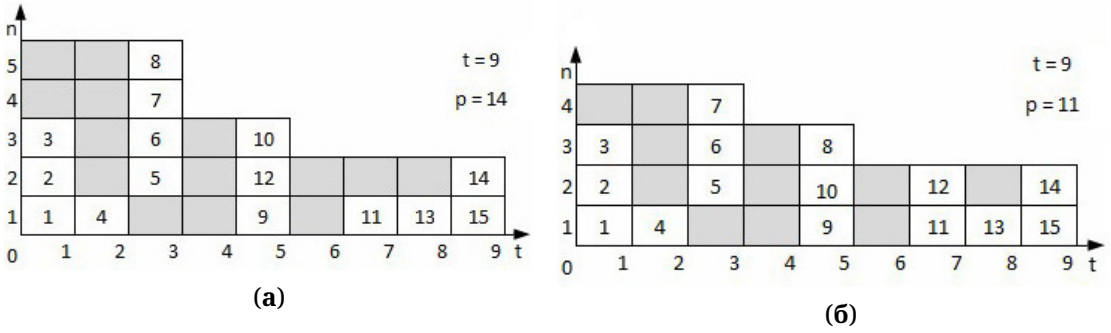


Рис. 3. Временная диаграмма алгоритма с учетом времени межпроцессорной передачи данных

В соответствии с данной диаграммой общее время работы алгоритма $t = 9$, число процессоров $n = 4$, суммарный объем простоев $p = 11$.

Полученное расписание лучше начального, так как позволяет использовать вычислительную систему с меньшим числом процессоров, сохранив при этом общее время выполнения алгоритма и сократив простои процессоров.

При оптимизации по объему коммуникаций без учета времени выполнения операций картина становится иной.

2.1. Обход графа по ширине с начала

В соответствии с этим методом для каждого яруса будут получены следующие группы вершин:

{1, 2, 3}
{4, 7, 5, 6}
{9, 10, 8}
{11, 12}
{13, 14}
{15}

Временная диаграмма после оптимизации информационного графа (рис. 1) по объему коммуникаций (в ширину с начала) показана на (рис. 4 а). В соответствии с данной диаграммой общее время работы алгоритма составит $t = 9$, число процессоров $n = 4$, суммарный объем простоев $p = 11$.

2.2. Обход графа по ширине с конца

В результате для алгоритма, соответствующего графу с рис. 1, будет получено следующее расписание:

{1, 2, 3}
{7, 4, 5, 6}
{10, 8, 9}
{11, 12}
{13, 14}
{15}

Временная диаграмма после оптимизации информационного графа (рис. 1) по объему коммуникаций (в ширину с конца) показана на (рис. 4 б). В соответствии с данной диаграммой общее время работы алгоритма составит $t = 8$, число процессоров $n = 4$, суммарный объем простоев $p = 9$.

2.3. Обход графа в глубину с начала

В соответствии с этим методом для графа с рис. 1 будут получены следующие группы для каждого яруса:

{1, 2, 3}
{7, 4, 5, 6}
{10, 9, 8}
{11, 12}
{13, 14}
{15}

Временная диаграмма после оптимизации информационного графа (рис. 1) по объему коммуникаций (в глубину с начала) показана на (рис. 4 в). В соответствии с данной диаграммой общее время работы алгоритма составит $t = 9$, число процессоров $n = 4$, суммарный объем простоев $p = 10$.

2.4. Обход графа в глубину с конца

В соответствии с этим методом для графа с рис. 1 будут получены следующие группы для каждого яруса:

{1, 2, 3}

{7, 4, 6, 5}

{10, 8, 9}

{11, 12}

{13, 14}

{15}

Временная диаграмма после оптимизации информационного графа (рис. 1) по объему коммуникаций (в глубину с конца) показана на (рис. 4 г). В соответствии с данной диаграммой общее время работы алгоритма составит $t = 8$, число процессоров $n = 4$, суммарный объем простоев $p = 9$.

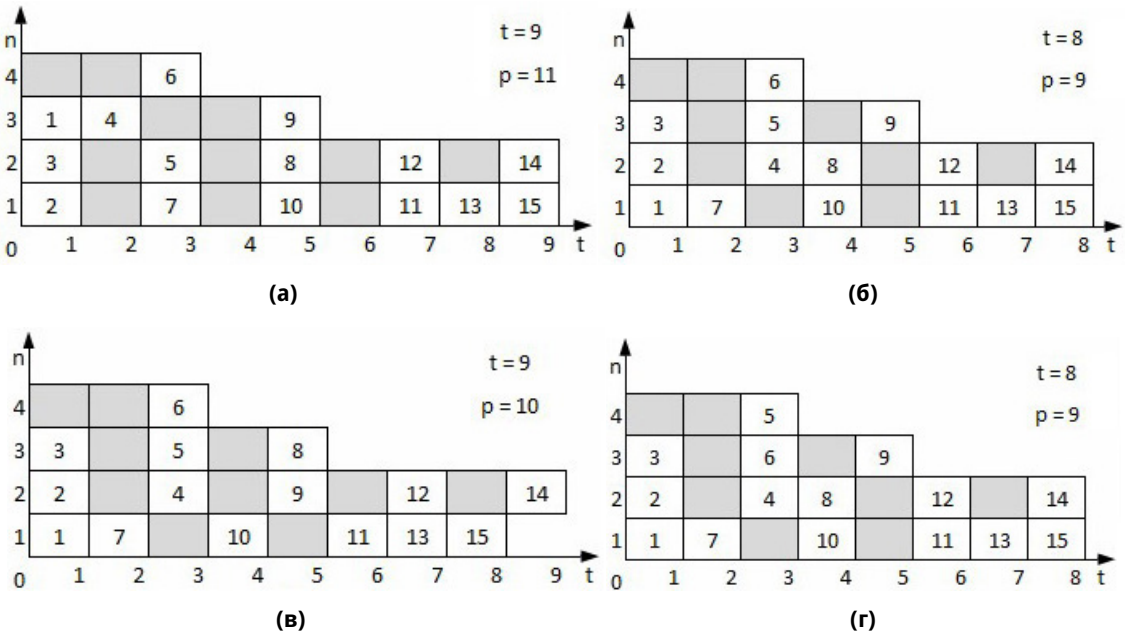


Рис. 4. Временная диаграмма оптимизированного алгоритма по объему коммуникаций

Время, затрачиваемое на передачу данных с одного процессора на другой, в лучшем случае уменьшилось почти в два раза и стало равным $p = 8$ ед. против $p = 14$ ед. начальных.

Суммарное время работы алгоритма уменьшилось с 9 до 8 в лучшем случае.

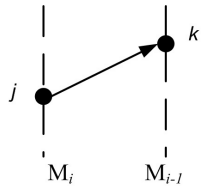
Несмотря на сходство методов, применение методов оптимизации с конца (в ширину и в глубину) являются более эффективными по сравнению с методами оптимизации с начала (в ширину и в глубину).

3. МЕТОД ОПТИМИЗАЦИИ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ПО ОБЪЁМУ МЕЖПРОЦЕССОРНЫХ КОММУНИКАЦИЙ С УЧЁТОМ ВРЕМЕНИ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ

В предлагаемом ниже методе после разбиения совокупности вершин на группы перестановка вершин начинается с последней группы вершин. Но далее будут рассмотрены и другие направления перестановки.

Метод оптимизации алгоритма по объему коммуникаций с учетом времени выполнения операций:

1. Разбить алгоритм на группы.
2. Процесс перестановки вершин начинается с последней группы. Начальные условия: количество всех групп N, номер очередной группы $k = M_{i-1}$.
3. Для M_{i_j}-ой вершины (i — номер яруса, j — номер вершины в ярусе) выбрать вершину k из M_{i-1}-ой группы, удовлетворяющей условиям:
   1) M_{i+1}, k имеет только одно входящее ребро, и это ребро (M_{i_j}, M_{i-1}, k)

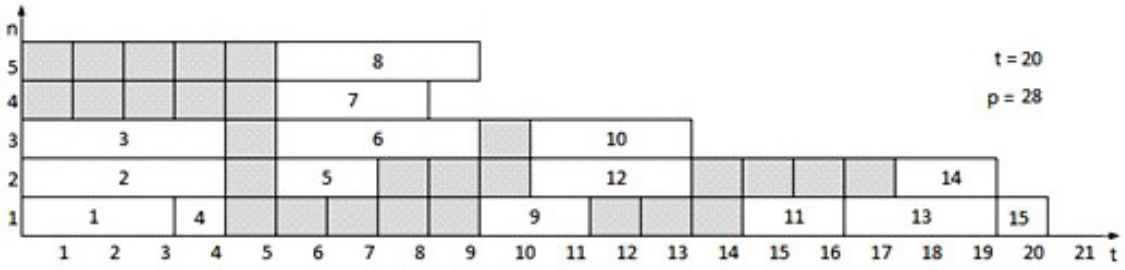
   2) k-я вершина не зафиксирована.
   3) Выполняется условие:
      $t_{2kj} > t_{2ks}$ для всех $s \neq j$,
      где t_2 – поздний срок окончания k-ой работы на j-м процессе. s — номер процесса.
      Если такая вершина k есть, то переставить её в M_{i-1}-ой группе на j-ое место и пометить как зафиксированную.
4. Повторить шаг 2 для всех вершин группы M_i.
5. Оставшиеся вершины группы M_i дополнить незафиксированными вершинами группы M_{i-1} в соответствии с правилом:
   Вершина M_{i-1}, k может быть переставлена в процессе j, если
   $t_{2kj} = \min t_{2ks}$,
   где $s \neq j$ — номер процесса, k — незафиксированная вершина.
6. Повторить шаги 2–4 для всех групп от N до 2.

Применим описанный метод к графу с рис. 1. Построим расписание выполнения алгоритма. Оно будет выглядеть, как показано на (рис. 5 а).

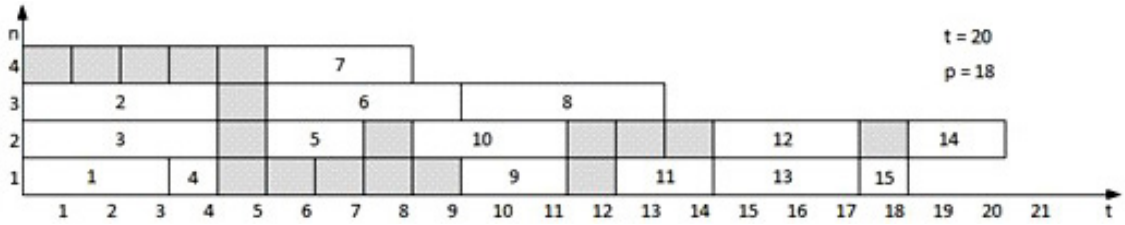
P_1: 1, 4, _, _, _, _, _, 9, _, _, _, 11, 13, 15;
P_2: 2, _, 5, _, _, _, 12, _, _, _, _, 14;
P_3: 3, _, 6, _, 10;
P_4: _, _, _, _, _, 7.
P_5: _, _, _, _, _, 8.

В соответствии с данным расписанием общее время работы алгоритма составит $t = 20$, число процессоров $n = 5$, суммарный объем простоев $p = 29$.

В соответствии с полученными после оптимизации по ширине группами (1) построим временную диаграмму алгоритма с учетом времени выполнения операций (рис. 5 б).



(а)



(б)

Рис. 5. Время выполнения алгоритма и количество простоев узлов в зависимости от направления оптимизации

Если в этом расписании (рис. 5 б) учесть время, затрачиваемое на передачу данных, то расписание изменится следующим образом:

P_1: 1, 4, _, _, _, _, _, 9, _, 11, 13, 15;

P_2: 3, _, 5, _, 10, _, _, 12, _, 14;

P_3: 2, _, 6, 8;

P_4: _, _, _, _, _, 7.

Общее время работы алгоритма при этом составит $t = 20$, число процессоров $n = 4$, суммарный объем простоев $p = 18$. Исследуем, как повлияет направление обхода графа в последнем описанном методе на время выполнения всего алгоритма.

3.1. Обход графа по ширине с начала

Временная диаграмма выполнения алгоритма представлена на рис. 6 а.

В соответствии с данной диаграммой общее время работы алгоритма составит $t = 19$, число процессоров $n = 4$, суммарный объем простоев $p = 16$.

3.2. Обход графа по ширине с конца

Временная диаграмма выполнения алгоритма представлена на рис. 6 б.

В соответствии с данной диаграммой общее время работы алгоритма $t = 17$, число процессоров $n = 4$, суммарный объем простоев $p = 11$.

3.3. Обход графа в глубину с начала

Временная диаграмма выполнения алгоритма представлена на рис. 6 в.

В соответствии с данной диаграммой общее время работы алгоритма $t = 17$, число процессоров $n = 4$, суммарный объем простоев $p = 16$.

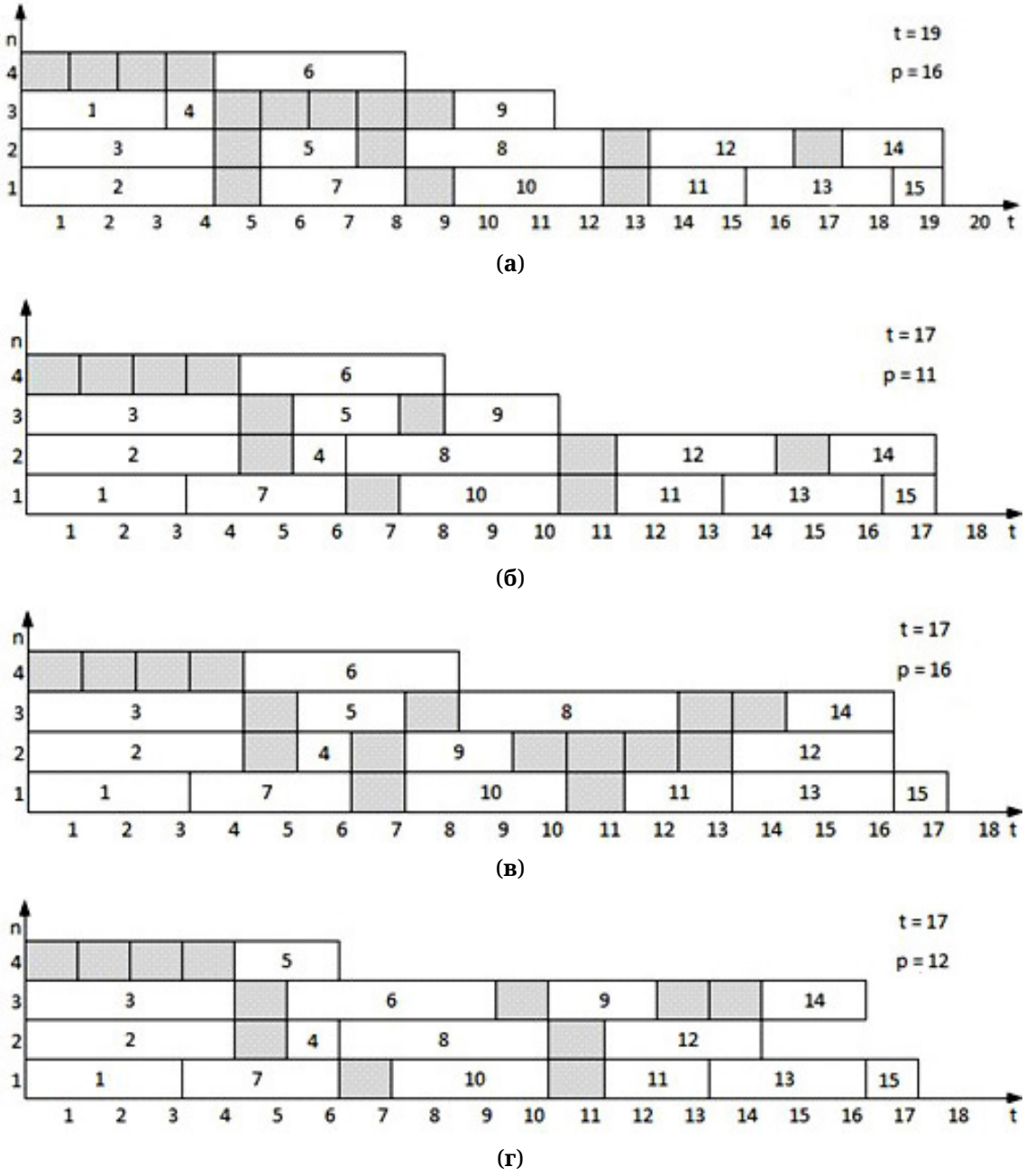


Рис. 6. Время выполнения алгоритма и количество простоев узлов в зависимости от направления оптимизации

3.4. Обход графа в глубину с конца

Временная диаграмма выполнения алгоритма представлена на рис. 6 г.

В соответствии с данной диаграммой общее время работы алгоритма $t = 17$, число процессоров $n = 4$, суммарный объем простоев $p = 12$.

Проанализировав объём коммуникаций, получим, что время, затрачиваемое на передачу данных с одного процессора на другой, уменьшилось в лучшем случае более чем в два раза и стало равным 11 ед., против 28 ед. начальных.

Суммарное время работы алгоритма уменьшилось с 20 до 17.

Проведенные исследования показали, что направление может существенно влиять на конечный результат. Несмотря на сходство методов, применение методов оптимизации с конца (в ширину и в глубину) является более эффективным по сравнению с методами оптимизации с начала (в ширину и в глубину).

4. АНАЛИЗ ВСЕХ НАПРАВЛЕНИЙ ПЕРЕБОРА ВЕРШИН В МЕТОДЕ ОПТИМИЗАЦИИ АЛГОРИТМА ПО ОБЪЕМУ КОММУНИКАЦИЙ

Программа протестирована на более 100 контрольных примерах.

Исходя из результатов тестирования программы, можно сделать вывод о том, что оптимизации в глубину и в ширину с конца являются самыми оптимальными, так как при оптимизации групп этими методами количество простоев на процессоре минимально.

В таблице 1 приведены данные о времени выполнения алгоритма, соответствующего заданному информационному графу, и простоях вычислительных узлов при выполнении этого алгоритма без учета времени выполнения самих операций.

Таблица 1. Время алгоритма и количество простоев узлов без учета времени выполнения операций

(Время (t), Простои (p))/Экспетимент (Е)	Е1(БезОпт)	Е1(ШсН)	Е2(ШсК)	Е3(ГсН)	Е4(ГсК)
t	9	8	9	8	9
p	14	9	11	9	10

Более наглядно увидеть время работы алгоритма после применения оптимизации по разным направлениям обхода графа можно на рис. 7 а. Из диаграммы видно, что оптимизация при обходе графа с конца является более эффективной по сравнению с методами оптимизации с начала (в ширину и в глубину).

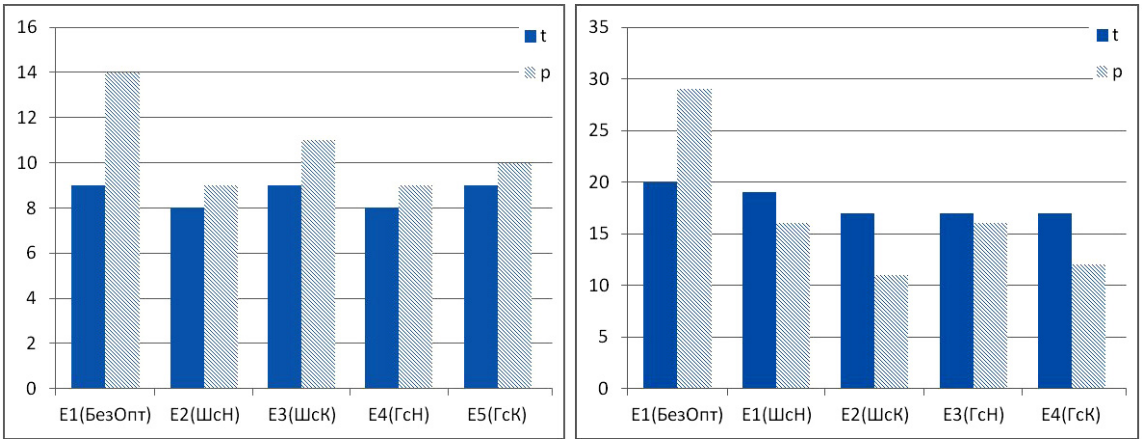
В таблице 2 приведены данные о времени выполнения этого же алгоритма, но с учетом времени выполнения операций.

Таблица 2. Время выполнения алгоритма и количество простоев с учетом времени выполнения операций

(Время (t), Простои (p))/Экспетимент (Е)	Е1(ДоОпт)	Е2(ШсН)	Е3(ШсК)	Е4(ГсН)	Е5(ГсК)
t	20	19	17	17	17
p	28	16	11	16	12

Из диаграммы (рис. 7 б), построенной по данным из таблицы 2, видно, что методы оптимизации по направлению обхода графа по ширине являются самыми оптимальными для случая, когда ведется учет времени выполнения операций.

Проведенные исследования и результаты многочисленных экспериментов показывают, что и с учётом и без учёта времени выполнения каждой операции, применение методов оптимизации с конца (в ширину и в глубину) являются более эффективными по сравнению с методами оптимизации с начала (в ширину и в глубину).



(а) **(б)**

Рис. 7. Время выполнения алгоритма и количество простоев узлов в зависимости от направления оптимизации

5. ЗАКЛЮЧЕНИЕ

В данной статье для наглядности все методы проиллюстрированы на одном примере. Но описанные методы были протестированы с помощью специальной программы на более чем 100 контрольных примерах. В результате чего установлено, что метод оптимизации в ШИРИНУ С КОНЦА является самым оптимальным и при оптимизации с учетом и без учета времени выполнения самих операций.

Методы оптимизации параллельного алгоритма по объёму коммуникаций позволяют уменьшить объём коммуникаций между процессорами и, соответственно, сократить время выполнения всего алгоритма.

Список литературы

1. *Abramov O. V., Katueva Ya.* Multivariant analysis and stochastic optimization using parallel processing techniques // Management problems. 2003. № 4. P. 11–15.
2. *Jordan H. F., Alaghband F.* Fundamentals of Parallel Processing. Pearson Education, Inc., Upper Saddle River, NJ, 2003. P. 578.
3. *Drake D. E., Hougardy S.* A linear-time approximation algorithm for weighted matchings in graphs // ACM Transactions on Algorithms. 2005. № 1. P. 107–122. doi: 10.1145/1077464.1077472
4. *Hu Chen.* MPIPP: An Automatic Profileguided Parallel Process Placement Toolset for SMP Clusters and Multiclusters / Hu.Chen // Proceedings of the 20th annual international conference on Supercomputing. New York, NY, USA. 2006. P. 353–360.
5. *Rauber N., Runger G.* Parallel Programming: for Multicore and Cluster Systems. / N. Rauber, G. Runger. Chemnitz, Germany: Springer, 2010. 450 p. doi: 10.1007/978-3-642-04818-0
6. *Gergel V. P., Fursov V. A.* Lectures of Parallel Programming: Proc. Benefit. Samara State Aerospace University Publishing House, 2009. 163 p.
7. *Voevodin V. V., Voevodin Vl. V.* Parallel computing. St. Petersburg: BHV-Petersburg, 2002. 608 p.
8. *Шичкина Ю. А.* Сокращение высоты информационного графа параллельных программ // Научно-технические ведомости СПбГПУ. 2009. № 3 (80). С. 148–152.

Поступила в редакцию 07.02.2018, окончательный вариант — 17.05.2018.

Computer tools in education, 2018
№ 3: 38–48
http://ipo.spb.ru/journal
doi:10.32603/2071-2340-3-38-48

COMPARATIVE ANALYSIS OF PARALLEL ALGORITHM'S OPTIMIZATION METHODS TAKING INTO CONSIDERATION OR IGNORING THE EXECUTION TIME OF OPERATIONS

Al-Mardi M. H. A.

Saint-Petersburg Electrotechnical University, Saint Petersburg, Russia

Abstract

In this paper, we propose an analysis of our (developed by us) methods for optimizing the parallel algorithm, taking into account and without taking into account the execution time of each operation. These methods can be applied on sequential algorithms in order to obtain their parallel analogue as well as on parallel algorithms in order to improve their quality. The proposed methods for optimizing the parallel algorithm can reduce the amount of communication between processors and, accordingly, reduce the execution time of the entire algorithm.

Keywords: *optimization, algorithm, information graph, sequence list, execution time, operation, process, processor, information dependence, unit of time.*

Citation: M. H. A. Al-Mardi, "Comparative Analysis of Parallel Algorithm's Optimization Methods Taking into Consideration or Ignoring the Execution Time of Operations," *Computer tools in education*, no. 3, pp. 38–48, 2018 (in Russian).

Received 07.02.2018, the final version — 17.05.2018.

Al-Mardi Mohammed Haidar Awadh, PhD student at department of Computer Science and Engineering–4, ETU «LETI»; 197376 , St. Petersburg, Russian Federation, ul. Professora Popova 5, building 2, Department of Computer Science and Engineering–4, almardi-md@mail.ru

Аль-Марди Мохаммед Хайдар Авадх, аспирант кафедры вычислительной техники–4 СПбГЭТУ «ЛЭТИ»; 197376, Санкт-Петербург, ул. Профессора Попова, д. 5, корп. 2, кафедра ВТ–4, almardi-md@mail.ru