УДК 681.3 DOI: 10.26102/2310-6018/2019.25.2.004

О.Ю. Лавлинская, В.В. Берников, О.Н. Григорова

РАСПАРАЛЛЕЛИВАНИЕ ВЫЧИСЛЕНИЙ ПОИСКА КРАТЧАЙШЕГО ПУТИ НА ОСНОВЕ ТЕХНОЛОГИИ OPENMP

Воронежский институт высоких технологий
Воронеж, Россия

Актуальность исследования обусловлена необходимостью решения задач распараллеливания вычислений для класса NP-полных задач, поскольку вычислительные мощности позволяют использовать параллельные потоки и снижать время вычислений и энергозатраты на получение сложного вычислительного результата с помощью программных технологий распараллеливания вычислений. В связи с этим, данная статья направлена на опубликование результатов исследования, основанных на анализе эмпирических данных, доказывающем эффективность параллельных вычислений для класса NP-полных задач. В статье проводится сравнение работы однопоточного приложения и многопоточного приложений, использующего технологию OpenMP, на примере алгоритма Флойда-Уоршалла поиска кратчайшего пути. В ходе эксперимента получены данные о скорости выполнения последовательного и параллельного алгоритмов. Сделаны выводы о том, что параллельный алгоритм эффективнее последовательного. При росте вычислительной мощности алгоритма эффективность параллельных вычислений возрастает. Эксперимент проводился для вычислений при разной мощности набора исходных данных, данные представлены в виде Таблиц и графиков. Дана оценка эффективности применения стандарта распараллеливания OpenMP. Материалы статьи представляют практическую значимость для студентов магистратуры, изучающих курс «Вычислительные системы», могут быть использованы для решения прикладных задач оптимизации.

Ключевые слова: OpenMP, распараллеливание, поиск путей, алгоритм Флойд-Уоршалла.

1. Введение

Стандарт OpenMP предоставляет инструментарий организации параллельных вычислений для вычислительных систем с несколькими узлами вычисления [1]. При решении NP-полных задач возникает задача поиска оптимального алгоритма решения и задача уменьшение времени за счет распараллеливания процессов (подзадач). В качестве примера рассмотрим задачу поиска кратчайшего пути с помощью алгоритма Флойда-Уоршалла [2].

Поиск кратчайшего пути – задача, имеющая прикладное значение: в логистике, в робототехнике (при поиске роботом оптимального маршрута), в структурно-топологическом анализе [3]. Математическая основа – дискретная математика, теория графов, где маршрут строится по графам.

Вычислительная техника бурно развивается, вычислительные системы наращивают количество ядер, содержащихся в центральных процессорах, а вместе с ними и количество потоков, способных обрабатывать информацию. Однако не все приложения могут корректно использовать все потоки, что выражается в том, что эти приложения используют один или несколько потоков, в то время как остальные простаивают. Таким образом, достаточно мощные системы могут проигрывать в производительности системам с меньшими количествами потоков и ядер, но с более высокой производительностью на ядро/поток.

Для исследования эффективности решения за счет распараллеливания процессов был использован алгоритм Флойда-Уоршалла, в качестве алгоритма, хорошо подходящего для процедуры разбиения на подзадачи. Разработана его программная реализация, затем произведено распараллеливание данного алгоритма для систем с общей памятью с использованием OpenMP и, в итоге, произведены вычислительные эксперименты на основании данных, сгенерированных заранее. Эти данные представляют информацию, записанную в специальном формате в текстовый файл, в виде разреженной матрицы (графа) [4] с различным количественным содержанием вершин и рёбер.

2. Материалы и методы

Рассмотрим взвешенный граф $G = (V, E)$, имеющий вершины V и ребра E ($|V| = n$, $|E| = m$). Веса обозначим $w_{ij} = w(v_i, v_j)$, $v_i, v_j \in V$. Пусть граф G - ориентированный (Рисунок 1).

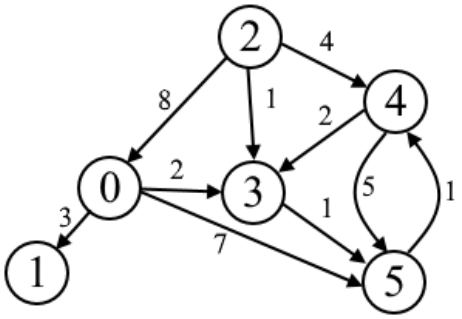


Рисунок 1 — Пример взвешенного ориентированного графа

Рассмотрим алгоритм поиска кратчайших путей, под названием «Алгоритм Флойда–Уоршалла», по имени авторов [5].

Составляется матрица смежности

$$A = (a_{ij}), 0 \le i, j \le n - 1 , \quad (1)$$

Задаются веса ребер графа

$$a_{i,j} = \begin{cases} w_{i,j}, \text{если } (v_i, v_j) \in E, \\ 0, \quad \text{если } i = j, \\ \infty, \quad \text{иначе} \end{cases} \tag{2}$$

Матрица смежности, соответствующая графу на Рисунке 1, приведена на Рисунок 2.

$$\begin{pmatrix} 0 & 3 & \infty & 2 & \infty & 7 \\ \infty & 0 & \infty & \infty & \infty & \infty \\ 8 & \infty & 0 & 1 & 4 & \infty \\ \infty & \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & \infty & 2 & 0 & 5 \\ \infty & \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

Рисунок 2 — Граф. Матрица смежности

Пусть путь $p = (v_0, v_1, \dots, v_i, v_j, \dots, v_{k-1})$. Пусть данный путь кратчайший, тогда пути $p_{0,i} = (v_0, \dots, v_i)$, $p_{i,j} = (v_i, v_j)$, $p_{j,k-1} = (v_j, \dots, v_{k-1})$ также будут кратчайшими.

Так как длина пути складывается из суммы длин его частей, то к пути p добавляем вершину v_k. Возможны два варианта:

Если длина (с учетом веса -стоимость) пути $p_{i,j}$ меньше стоимости пути $p'_{i,j} = (v_i, v_k, v_j)$, то кратчайший путь не изменится.

Если длина (стоимость) $p_{i,j}$ больше длины $p'_{i,j} = (v_i, v_k, v_j)$, тогда кратчайшим будет путь $p' = (v_0, v_1, \dots, v_i, v_k, v_j, \dots, v_{k-1})$.

Изменим нумерацию вершин на «0 до $n - 1$".

Определим матрицу $D = (d_{i,j})$ расстояний между вершинами без промежуточных вершин значения элементов которой $d_{i,j}$, $0 \le i, j \le n - 1$ совпадают с весами $w_{i,j}$ перехода из вершины i в вершину j. Если ребро $e_{i,j}$ отсутствует, то $d_{i,j} = \infty$ кроме того $d_{i,i} = 0$.

$$d_{i,j} = \begin{cases} w_{i,j}, \text{если } e_{i,j} \in E, \\ 0, \quad \text{если } i = j, \\ \infty, \quad \text{иначе} \end{cases} \tag{3}$$

Таким образом, исходная матрица расстояний совпадает с матрицей смежности исследуемого графа.

Если $d_{i,j}^{k}$ – это длина искомого пути через промежуточную вершину k, то D^{k} – это матрица размера $n \times n$, где элемент (i,j) совпадает с $d_{i,j}^{k}$. Для обхода всех $k-$ых вершин последовательно принимающей значения $0,1,\dots,n-1$ вычислим D^{k-1} элементы матрицы D^{k}, применяя рекуррентное соотношение

$$d_{i,j}^{k} = \begin{cases} \min\left(d_{i,j}, d_{i,k} + d_{k,j}\right), & \text{если } k = 0, \\ \min\left(d_{i,j}^{k-1}, d_{i,k}^{k-1} + d_{k,j}^{k-1}\right), & \text{если } k > 0, \end{cases} \tag{4}$$

Матрица $D^{n-1} = \left(d_{i,j}^{n-1}\right)$, будет матрицей, содержащей длины кратчайших путей для всех пар вершин $i,j \in V$. Для поиска удобно использовать простые рекуррентные соотношения

$$P_{i,j} = \begin{cases} \text{NULL}, & \text{если } w_{i,j} = \infty, \\ i, & \text{если } i = j, \\ i, & \text{если } w_{i,j} < \infty. \end{cases}$$

$$P_{i,j}^{0} = \begin{cases} P_{i,j}, & \text{если } d_{i,j} \le d_{i,k} + d_{k,j}, \\ P_{k,j}, & \text{если } d_{i,j} > d_{i,k} + d_{k,j}. \end{cases} \quad (k = 0)$$

$$P_{i,j}^{k} = \begin{cases} P_{i,j}^{k-1}, & \text{если } d_{i,j}^{k-1} \le d_{i,k}^{k-1} + d_{k,j}^{k-1}, \\ P_{k,j}^{k-1}, & \text{если } d_{i,j}^{k-1} > d_{i,k}^{k-1} + d_{k,j}^{k-1}, \end{cases} \quad (k > 0) \tag{5}$$

Сложность последовательного алгоритма Флойда–Уоршалла имеет порядок n^3, что означает при увеличении количества вершин в графе в 2 раза увеличение времени работы алгоритма в 8 раз и так далее.

Далее в работе рассматривается параллельная реализация алгоритма Флойда–Уоршалла на основе технологии OpenMP, с учетом нюансов, описанных в [6].

Идея параллельного алгоритма Флойда–Уоршалла состоит в том, что операция рекуррентного поиска кратчайших путей осуществляется независимо.

В соответствии с общей схемой алгоритма Флойда–Уоршалла необходимо $n-$раз выполнить однотипную операцию, обновляющую матрицу кратчайших путей.

3. Результаты

Для организации параллельных вычислений необходимо, чтобы в одном потоке при рекуррентном соотношении соблюдалось постоянство сохранения данных для одного потока.

Обратим внимание на то, что на $k-$ой итерации алгоритма Флойда–Уоршалла не происходит изменения элементов $k-$ой строки и $k-$ого столбца ни для одной пары индексов (i, j).

Действительно, $d_{i,k}^{k} = d_{i,k}^{k-1}$ и $d_{k,j}^{k} = d_{k,j}^{k-1}$ согласно рекуррентному соотношению (5), так как вершина k не может выступать в качестве промежуточных в любых кратчайших путях, которые начинаются или заканчиваются в самой вершине k (Рисунок 3).

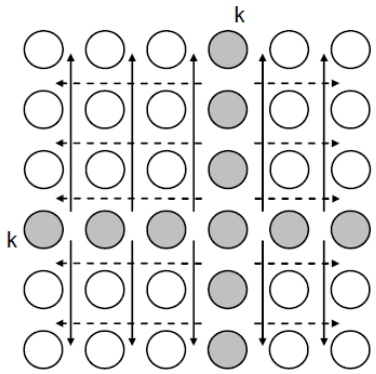


Рисунок 3 — Выделение базовых процессов (подзадач) (на k-ой итерации показаны стрелками направления информационного обмена)

Если $i = k$ то:

$$d_{k,j}^{k} = \min\left(d_{k,j}^{k-1}, d_{k,k}^{k-1} + d_{k,j}^{k-1}\right),$$

При этом $d_{k,k}^{k-1} = 0$.

Тогда для $j = k$ получим:

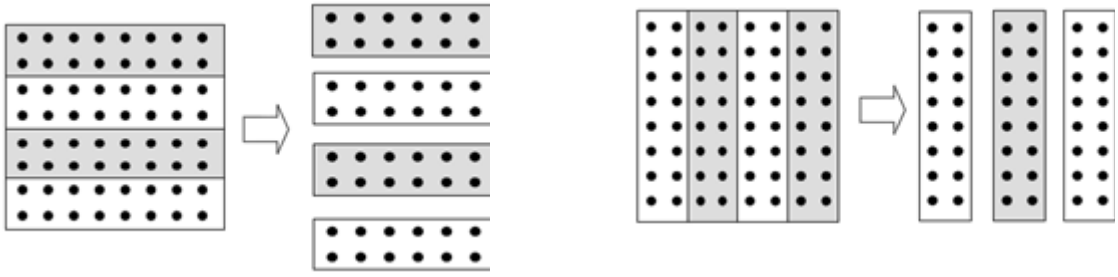
$$d_{i,k}^{k} = \min\left(d_{i,k}^{k-1}, d_{i,k}^{k-1} + d_{k,k}^{k-1}\right),$$

что доказывает неизменность $d_{i,k}^{k}$.

Таким образом, константность значений элементов $k-$ой строки и $k-$ого столбца на итерации k доказана.

Если математически есть предпосылки к организации параллельных вычислений, то остается создать программную реализацию.

В [1] предложен подход к организации параллельных вычислений, представленный на Рисунке 4 а, б. В соответствии с данным подходом выделяются горизонтальные или вертикальные «ленты» непрерывности.



а) б)

Рисунок 4 — Представление непрерывности в виде процессов
а) по горизонтали и б) по вертикали

Для удобства восприятия выберем горизонтальный способ представления последовательности, по аналогии с организацией оперативной памяти компьютера, и будем рассматривать далее только разбиение матрицы на горизонтальные полосы.

Следует отметить, что при таком способе разбиения данных на каждой итерации алгоритма Флойда–Уоршалла, отдельным потокам требуется строка матрицы, номер которой совпадает с номером итерации.

Одним из преимуществ разработки алгоритмов для систем с общей памятью является отсутствие необходимости организации обмена данными между потоками и выделения памяти для хранения данных.

После разделения матрицы на горизонтальные полосы, вычисления над ними производятся в произвольном порядке.

Алгоритм продолжает свою работу до тех пор, пока в качестве ведущей вершины k не побывают все вершины из множества вершин V.

Вычислительные эксперименты проводились с использованием следующей инфраструктуры (Таблица 1).

Таблица 1 — Тестовая инфраструктура

Процессор	1 двухъядерный процессор Intel Celeron B815 (1.6GHz)
Память	2 Gb
Операционная система	Microsoft Windows 7

Среда разработки	Microsoft Visual Studio 2012: Version 11.0.50727.1 Microsoft ® C/C++ Compiler Driver Version 17.0.50727.1

4. Результаты

Результаты проведенного эксперимента оформим в виде Таблицы 2, в которой приведем данные по оценке ускорения вычисления в зависимости от количества вершин на примере последовательного и параллельного вычисления алгоритма.

Таблица 2 — Результаты сравнительного анализа последовательного и параллельного вычисления для алгоритма Флойда-Уоршалла

Количество вершин	Последовательный алгоритм	Параллельный алгоритм	Ускорение
1000	5,0044	3,6613	1,3668
2000	41,6662	28,0307	1,4864
3000	150,9231	91,7306	1,6452
4000	411,1675	244,0871	1,6845
5000	879,9519	520,7946	1,6896

На Рисунке 5 показано увеличение скорости работы параллельного алгоритма по сравнению с последовательным.

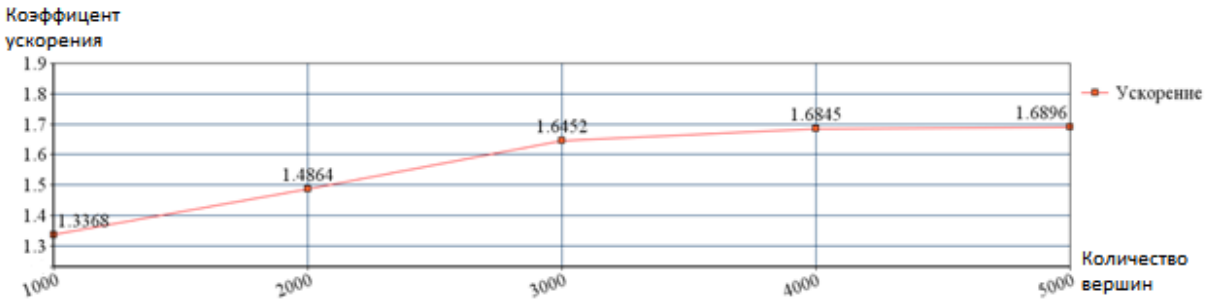


Рисунок 5 — График сравнения скорости работы последовательного и параллельного алгоритма

На Рисунке 6 показано сравнение времени работы последовательного и параллельного алгоритмов

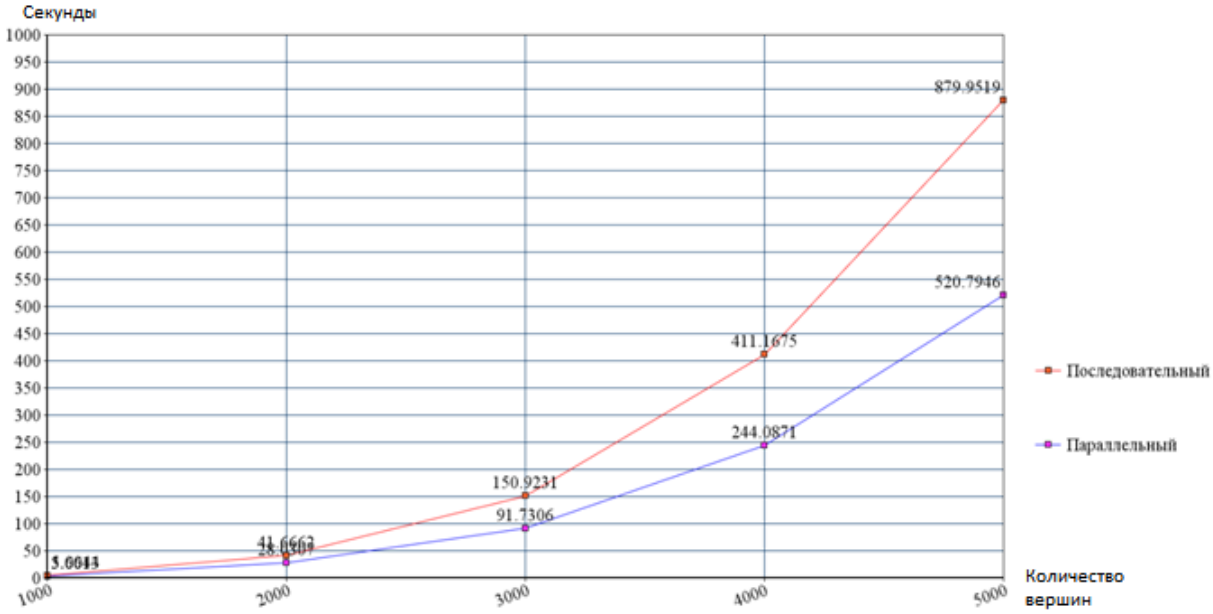


Рисунок 6 — График сравнения скорости работы последовательного и параллельного алгоритма

Из полученных результатов мы можем сделать вывод, что параллельный алгоритм выполняется примерно в 1,5 раза быстрее последовательного (для 3000 вершин в графе), а с увеличением количества вершин эта разница увеличивается.

5. Заключение

В данной статье рассматривается задача поиска кратчайшего пути в графе с помощью алгоритма Флойда-Уоршалла, для которого выполняется последовательная и параллельная реализации, поскольку использование в программировании параллельных вычислений позволяет сокращает время выполнения расчета.

Для получения данного результата были произведены следующие действия:

1. Изучен последовательный алгоритм Флойда-Уоршалла.
2. Выполнена программная реализация последовательной версии и параллельной версии алгоритма Флойда-Уоршалла для систем с общей памятью с использованием OpenMP.
3. Описаны программная инфраструктура и формат хранения графа, моделирующего карту дорог Рима [5], для выполнения экспериментов.
4. Проведены вычислительные эксперименты, в результате которых установлено, что распараллеленный алгоритм Флойда-Уоршалла позволяет сократить время вычислений до полутора раз.

ЛИТЕРАТУРА

1. Гергель В. П. Параллельные вычисления: технологии и численные методы: Учебное пособие в 4 томах \\ В. П. Гергель и др. Н. Новгород: Издательство Нижегородского госуниверситета, – 2013. – 239 с.
2. Левитин А. В. Глава 11. Преодоление ограничений: Метод деления пополам // Алгоритмы. Введение в разработку и анализ — М.: Вильямс, 2006. — С. 349–353. — 576 с.
3. Лавлинская О. Ю Применение теории графов в структурно-топологическом анализе информационных систем \\ О. Ю. Лавлинская, Т. В. Курченкова. – Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. – 2017. – № 23 (272). – С. 105-112.
4. Томас Х. Кормен и др. Глава 34. NP-полнота // Алгоритмы: построение и анализ. – 2-е изд. – М.: «Вильямс», – 2006. – С. 1296.
5. Roy, Bernard (1959). “Transitivité et connexité”. C. R. Acad. Sci. Paris (англ.)русск. 249: 216 – 218.
6. Карпов А. Отладка и оптимизация многопоточных OPENMP-программ \\ RSDN Magazine. — 2008. — № 4. — С. 32-36.

O.Y. Lavlinskaya, V.V. Bernikov, O.N. Grigorova

OPENMP PARALLEL CALCULATIONS IN ALGORITHMS FOR SOLVING SHORTEST PATHS PROBLEM

Voronezh institute of high technologies
Voronezh, Russia

The relevance of the study is due to the need to solve the problems of parallelization of calculations for the class of NP-complete problems, since computing power allows the use of parallel flows and reduce the computation time and energy consumption to obtain a complex computational result using software technologies of parallelization of calculations. In this regard, this article aims to publish the results of the study, based on the analysis of empirical data, proving the effectiveness of parallel calculations for the class of NP-complete problems. The article compares the work of a single-threaded application and multithreaded applications using OpenMP technology, on the example of the Floyd-Warshall Algorithm for finding the shortest path. During the experiment, data on the speed of serial and parallel algorithms were obtained. It is concluded that the parallel algorithm is more efficient than the serial one. With the growth of the computational power of the algorithm, the efficiency of parallel computing increases. The experiment was carried out for calculations at different power of a set of initial data, the data are presented in the form of tables and graphs. Evaluate the effectiveness of the

use of standard parallelization and OpenMP. The materials of the article are of practical importance for master's students studying the course "Computer systems", can be used to solve applied optimization problems.

Keywords: OpenMP, parallelization, search of ways, Floyd- Warshall algorithm

REFERENCES

1. Gergel' V. P. Parallel'nyye vychisleniya: tekhnologii i chislennyye metody: Uchebnoye posobiye v 4 tomakh \\ V. P. Gergel' i dr. N. Novgorod: Izdatel'stvo Nizhegorodskogo gosuniversiteta, - 2013. - 239 s.
2. Levitin A. V. Glava 11. Preodoleniye ogranicheniy: Metod deleniya popolam // Algoritmy. Vvedeniye v razrabotku i analiz - M .: Vil'yams, 2006. - S. 349-353. - 576 s.
3. Lavlinskaya O. Primeneniye teorii grafov v strukturno-topologicheskom analize informatsionnykh sistem \\ O. YU. Lavlinskaya, T. V. Kurchenkova. - Nauchnyye vedomosti Belgorodskogo gosudarstvennogo universiteta. Seriya: Ekonomika. Informatika. - 2017. - № 23 (272). - S. 105-112.
4. Tomas KH. Kormen i dr. Glava 34. NP-polnota // Algoritmy: postroyeniye i analiz. - 2-ye izd. - M .: «Vil'yams», - 2006. - S. 1296.
5. Roy, Bernard (1959). «Tranzitiv i svyaz'». C. R. Acad. Sci. Parizh (angl.) Russk. 249: 216 - 218. 6. Karpov A. Otladka i optimizatsiya mnogopotochnykh OPENMP-programm \\ RSDN Magazine. - 2008. - № 4. - S. 32-36.