электронное научно-техническое издание
НАУКА и ОБРАЗОВАНИЕ
Эл № ФС 77 - 30569. Государственная регистрация №0421100025. ISSN 1994-0408

Разработка параллельного алгоритма построения опорного плана транспортной задачи

 # 05, май 2011

автор: Аль-хулайди А. А.

УДК 004.032.24

ДГТУ – г. Ростов-на-Дону
admin@abdulmajed.8m.net
alkhulaidi_2006@hotmail.com

Введение

Рассматриваемая задача может найти применение в различных областях:

- при решении задачи распределения ресурсов между вычислительными задачами или устройствами;
- в экономике, при планировании транспортного комплекса;
- в учебном процессе.

Таким образом, можно говорить об актуальности данной задачи.

Использование параллельного алгоритма, оптимизированного для работы в многопроцессорной среде, позволяет более оптимально использовать вычислительные мощности кластера.

Подходы к решению транспортной задачи с помощью параллельных алгоритмов изложены, например, в работе [1]. В работе [2] приведены экспериментальные данные, полученные при выполнении параллельных алгоритмов нахождения опорного плана транспортной задачи на кластере, однако отсутствует сколько-нибудь подробное описание применявшихся при этом методов. Монография [3] содержит подходы к распараллеливанию методов решения задачи целочисленной оптимизации, которые могут быть применены и при решении транспортной задачи.

В данной статье подробно описан параллельный алгоритм нахождения опорного плана транспортной задачи, приводится блок-схема и данные о его производительности, полученные экспериментальным путём.

1. Постановка задачи

Транспортная задача является специальным типом задач линейного программирования. Математическая постановка этой задачи имеет вид [4]:

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}X_{ij};$$

$$\sum_{i=1}^{m} X_{ij} = b_j, \qquad j = 1,2,\ldots, n; \qquad (1)$$

$$\sum_{j=1}^{n} X_{ij} = a_i, \qquad i = 1,2,\ldots, m;$$

$x_{ij} \geq 0$, $i = 1, 2, \ldots, m$, j = 1, 2, …, n.

Здесь X_{ij} – объем, c_{ij} – тариф поставки продукции от i-го поставщика к j-му потребителю, b_j – потребности потребителей в продукции, a_i – запасы продукции у поставщиков.

Видно, что задача (1) является задачей линейного программирования со специальной матрицей. В задаче (1) имеется $(m \cdot n)$ неизвестных X_{ij} и $(m + n)$ уравнений.

Решение транспортной задачи называется *оптимальным планом перевозок (поставок) продукции*.

Метод распределения ресурсов состоит из двух этапов:

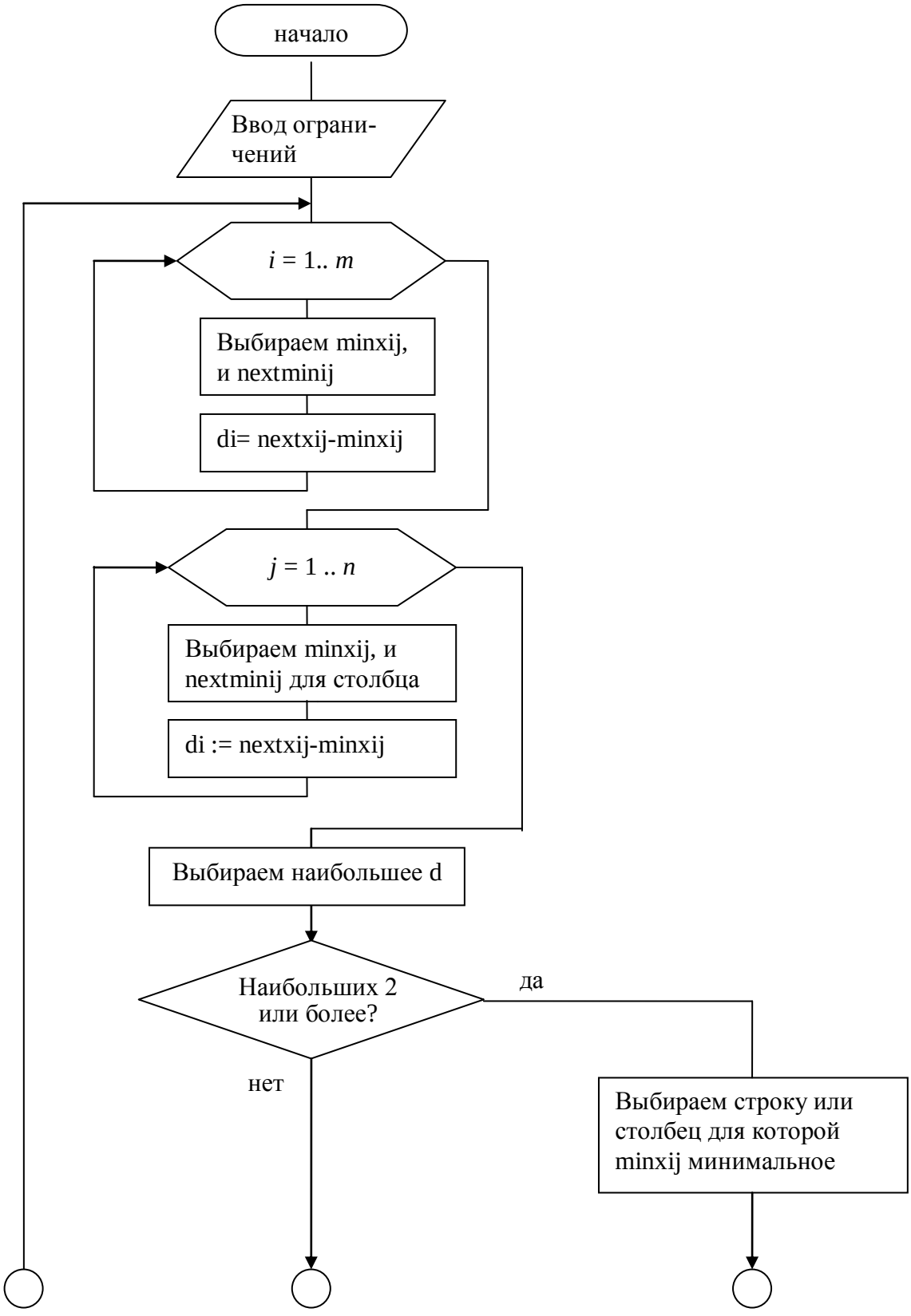
1) построение опорного плана;
2) поиск оптимального плана.

Для получения опорного плана используются различные алгоритмы, такие как метод минимального элемента; метод северо-западного угла; метод штрафов [5]. Наиболее удобным для распараллеливания является метод штрафов, который позволяет независимо выбрать план для каждого столбца и строки. Причём основная часть операций может быть выполнена параллельно.

Последовательный алгоритм решения транспортной задачи методом Фогеля (штрафов) для получения опорного плана выглядит следующим образом.

1) Штрафы для строки или столбца представляют положительную разницу между минимальным элементом строки (столбца) и следующим за ним по величине минимальным элементом строки или столбца.
2) Расчет штрафов для всех строк и столбцов матрицы.
3) Формирование опорного плана начинается с минимального элемента строки или столбца с максимальным штрафом.
4) Выбор элементов плана X и коррекция векторов потребления и постановок осуществляется, как рассмотрено выше (шаги 1, 2).
5) Строка или столбец матрицы С, которым соответствует нулевое значение потребности или поставки, в дальнейшем не участвуют в формировании штрафов. Процесс продолжается до тех пор, пока не обнулятся все вектора.

На рис. 1 представлена схема последовательного алгоритма нахождения опорного плана методом штрафов.



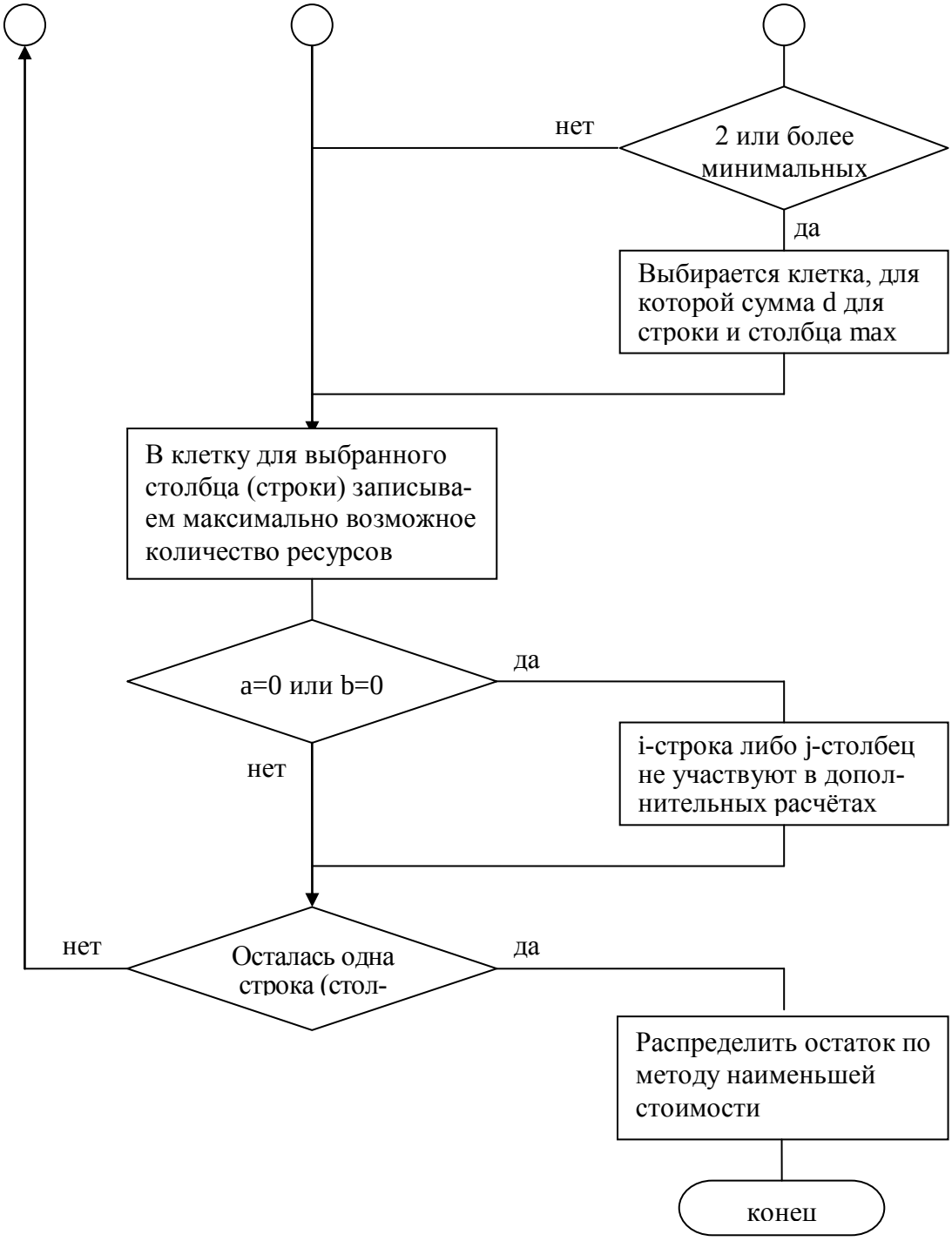


Рис. 1. Схема последовательный алгоритм нахождения опорного плана методом штрафов.

Решение транспортной задачи большой размерности при помощи последовательной модели вычислений требует значительных временных затрат. В связи с этим актуальной является разработка соответствующих параллельных алгоритмов, позволяющих сократить время поиска решения.

Так как транспортная задача имеет экономический подтекст, она является актуальной для различного масштаба производственных и торговых предприятий. Наиболее доступным видом параллельных вычислительных систем для таких предприятий представляется вычислительный кластер, представляющий собой систему с распределённой памятью.

Ставится следующая задача. Модифицировать приведенный выше последовательный алгоритм метода Фогеля для выполнения на вычислительном кластере с использованием в качестве коммуникационной библиотеки – *MPI* (*Message Passing Interface*).

2. Разработка модификации метода Фогеля для параллельных систем

Пусть u — вектор потребностей потребителей, v — вектор мощностей поставщиков, z — матрица стоимости перевозок, x — матрица решения (поставок).

Параллельный алгоритм для нахождения опорного решения транспортной задачи выглядит следующим образом.

1) Для каждой строки и столбца матрицы стоимости перевозок определяем разницу между наименьшим значением стоимости и ближайшим к нему (т. н. *штраф*). Поиск штрафа по каждой строке и столбцу оформляется в виде отдельной задачи и назначается свободному процессору.

2) Во всех строках и столбцах с наибольшим штрафом находим ячейки с наименьшим минимальным значением стоимости (не зачеркнутые ранее). Выполнение этого шага также распределяется между свободными процессами.

3) Максимизируем поставку в ячейку $x[i,j]$, выбрав минимальное значение из потребности потребителя $u[i]$ и мощности поставщика $v[j]$.

4) Если мощность поставщика полностью реализована или потребность потребителя полностью удовлетворена, вычеркиваем соответствующую строку или столбец.

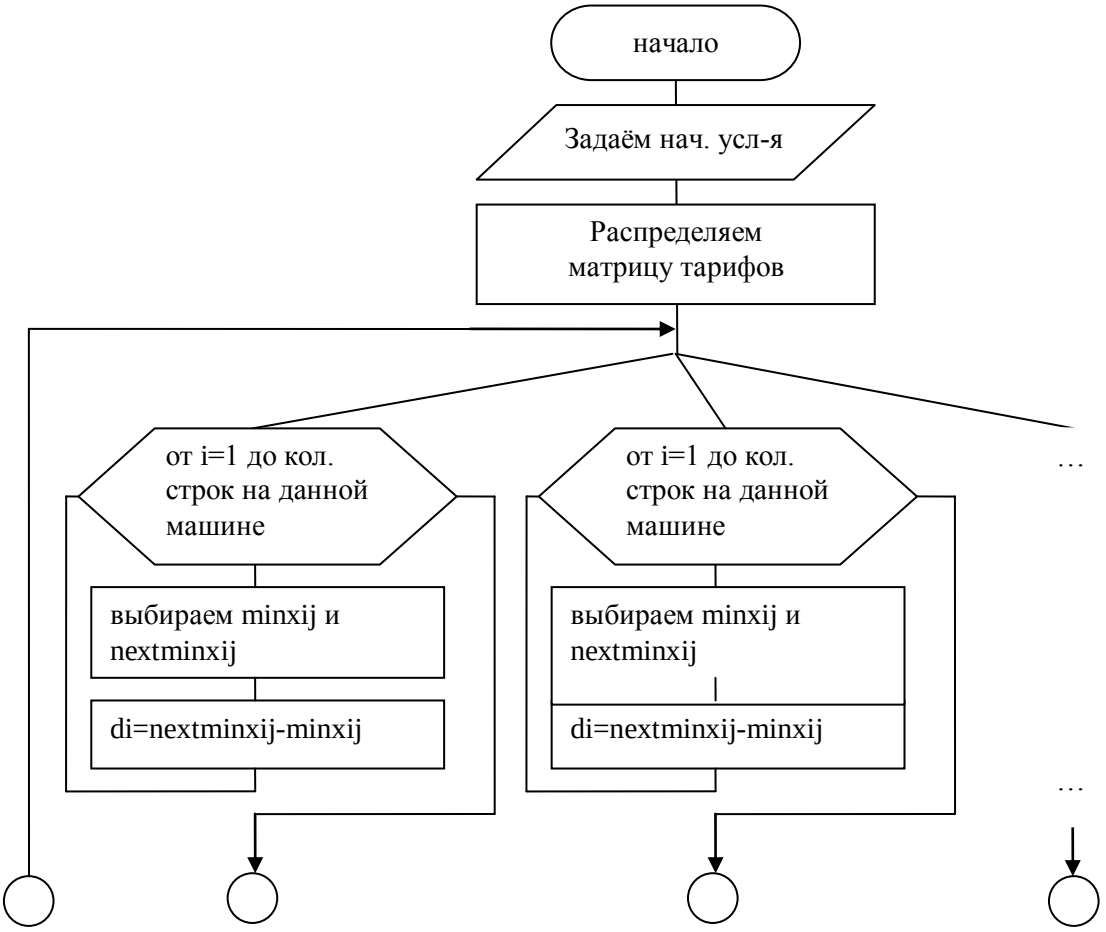
5) Если не все потребности (мощности) задействованы, то повторяем алгоритм.

Параллельная модификация алгоритма использует тот факт, что подсчет штрафов и поиск минимального элемента в пунктах 1, 2 поддаются параллелизации за счёт разделения задач по поиску в разных строках и столбцах по различным процессорам кластера. На рис. 2 представлена схема параллельного алгоритма нахождения опорного плана методом Фогеля.

3. Оценка времени выполнения параллельного алгоритма

Метод штрафов заключается в том, что для каждого столбца и каждой строки матрицы отыскивается минимальный и следующий за ним по значению элемент. Разница между ними называется штрафом строки или столбца. Среди всех штрафов выбирается тот, который имеет максимальное значение. Минимальный элемент строки или столбца включается в опорный план, а соответствующая строка или столбец исключается из дальнейшего рассмотрения. Эти действия повторяется до тех пор, пока не будут исключены все строки и столбцы. Совокупность помеченных на каждой итерации элементов образуют опорный план. Последовательность работ на одной итерации приведена на рис. 3.

Для определённости будем считать, что задана квадратная матрица ресурсов (m*m), а число обрабатывающих процессорных узлов (процессоров) в параллельной системе равно q, причём $q \le 2m$. Все данные задачи и программа управлением решения находятся управляющем узле (сервере) *C*.



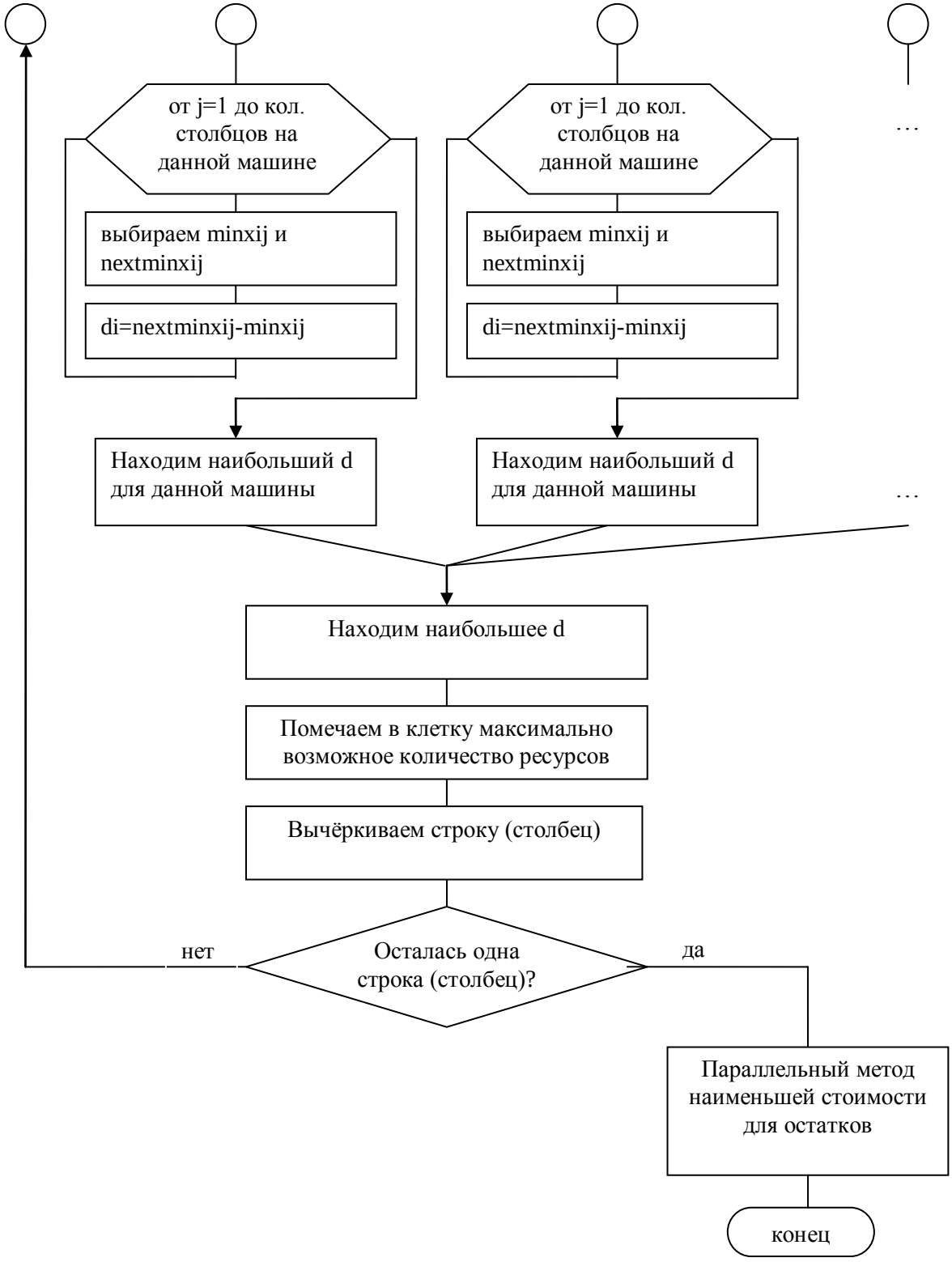


Рис. 2. Схема параллельного алгоритма нахождения опорного плана методом Фогеля

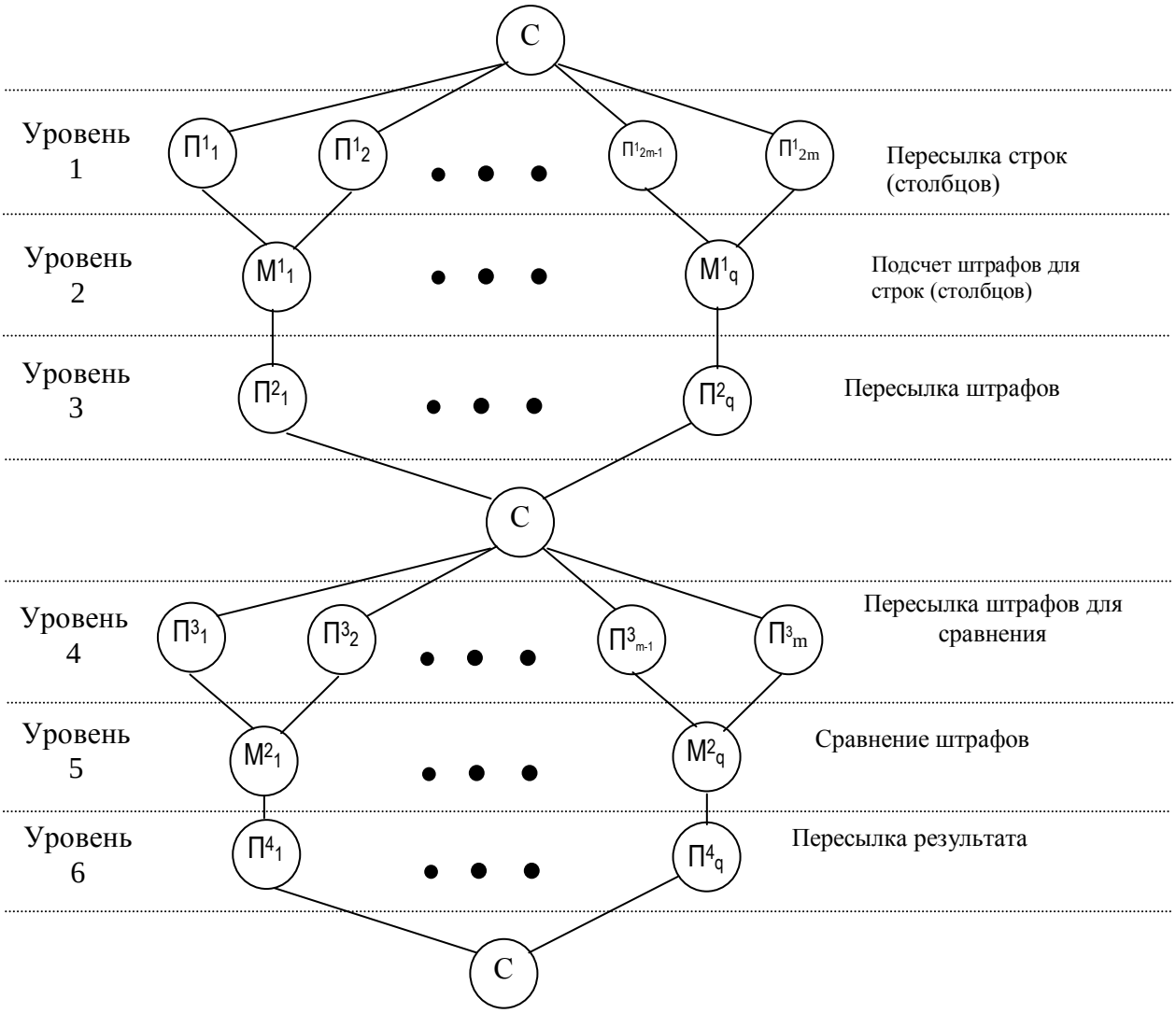


Рис. 3. Схема одного этапа выполнения работ для получения опорного плана методом штрафов

На уровне 1 сервер пересылает строки (столбцы) на процессоры. Число строк на каждом процессоре зависит от числа узлов. На уровне 2 процессор обрабатывает полученные строки, осуществляя поиск максимального штрафа среди всех строк, переданных ему сервером. На уровне 3 процессоры пересылают максимальный штраф и координаты строки (столбца), в которой он был получен, на сервер. На уровне 4 сервер распределяет полученные штрафы по процессорам для определения среди них максимального. На уровне 5 происходит передача результата поиска максимального штрафа на сервер. Сервер вычёркивает ту строку (столбец) в которой определён максимальный штраф и организует следующую итерацию.

Для расчёта числа операций при получении опорного плана методом штрафов обозначим $T_{П}^{1,j}$ – время пересылки матрицы по строкам (столбцам) с сервера на каждый из процессоров; $j = \overline{1,(m-1)}$ – номер итерации метода штрафов.

Время пересылки матрицы можно выразить через время пересылки одного элемента (элементарная пересылка). Легко видеть, что справедливы следующие выражения:

$$T_{П}^{1,j} = t_{эл.пер.} \times \left[m^2 - m \times (j-1) \right],$$

если вычёркиваются только строки (или только столбцы); $t_{эл.пер.}$ – время на пересылку одного элемента;

$$T_{П}^{1,j} = t_{эл.пер.} \times \left[m^2 - (j-1) \times \left(m - \frac{j-1}{2} \right) \right],$$

если вычёркиваются попеременно строка затем столбец.

Отсюда следует, что среднее значение величины $T_{П}^{1,j}$ равно

$$T_{ПСр}^{1,j} = t_{эл.пер.} \times \left[m^2 - (j-1) \times \left(m - \frac{j-1}{4} \right) \right].$$

Время $T_{О}^{1,j}$ обработки строки (столбца) на одном процессоре можно выразить через элементарные операции сравнения и представить в виде

$$T_{О}^{1,j} = [(m-j-1) + (m-j-2) + 1] t_{эл.обр.} = 2(m-j-1) t_{эл.обр.},$$

где $t_{эл.обр.}$ – время выполнения одной операции сравнения при поиске минимального и следующего за ним элемента.

Время, затрачиваемое на одном шаге для поиска штрафа в каждой строке (столбце) по всей матрице ресурсов для q узлов обработки, можно записать в виде

$$\begin{cases} \left(\left[\dfrac{2m-j+1}{q} \right] + 1 \right) T_{О}^{1,j}, \text{ если } q \le (2m-j+1), \\ T_{О}^{1,j}, \text{ если } q > (2m-j+1) \end{cases}.$$

Данная функция является дискретной. График этой функции представлен на рис. 4 (принято, что $q = \overline{1,20}$, $m = 10$, $j = \overline{1,2m}$ и $T_{О}^{1,j} = 1$). Для упрощения расчётов аппроксимируем эту функцию как функцию двух переменных q и $(2m\text{-}j + 1)$:

$$-45 \cdot 10^{-6} q^2 (2m-j+1)^2 + 9 \cdot 10^{-4} q (2m-j+1)^2 - 15 \cdot 10^{-4} (2m-j+1)^2 +$$
$$+ (4 \cdot 10^{-3} q^2 - 0{,}1163q + 0{,}768)(2m-j+1) - 36 \cdot 10^{-4} q^2 + 0{,}12q + 87 \cdot 10^{-4}.$$

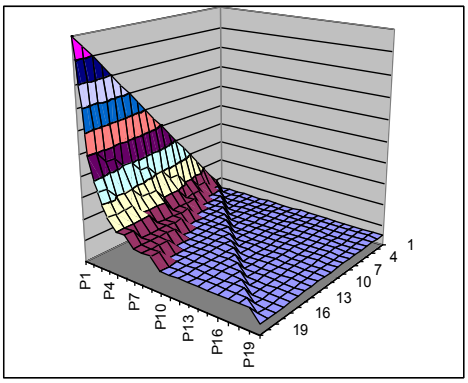


Рис. 4. График зависимости времени поиска штрафа от числа строк и узлов обработки

Аппроксимирующая функция имеет вид, представленный на рис. 5.

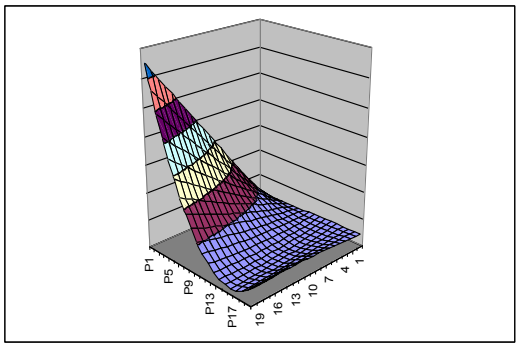


Рис. 5. График аппроксимирующей зависимости времени поиска

Введем следующие обозначения:

$T_{П}^{2} = 3 * t_{эл.пер.}$ - время пересылки результата обработки одной строки (столбца) на сервер с обрабатывающего процессора;

$T_{П}^{3}$ - время пересылки величины штрафа и индексов минимума строки (столбца) с сервера на процессор.

Время $T_{П}^{3}$ зависит от стратегии поиска максимального штрафа среди всех найденных. Обычно все полученные штрафы распределяются равномерно на q имеющихся процессоров. Поэтому это время через элементарные операции можно записать в виде

$$T_{П}^{3} = \begin{cases} 3t_{эл.пер.} \cdot (2m - j + 1) \sum_{i=0}^{[\log_q (2m-j+1)]-1} \frac{1}{q^i}, \text{ если } 2m - j + 1 > q, \\ 3t_{эл.пер.} \cdot (2m - j + 1) \sum_{i=0}^{\log_q (2m-j+1)} \frac{1}{2^i}, \text{ если } 2m - j + 1 \le q. \end{cases}$$

Для упрощения расчётов аппроксимируем данную формулу как функцию от переменной q:

$$T_{П}^{3} \approx 3t_{эл.пер.} \cdot (2m - j + 1)(-0{,}023q^2 + 2{,}2843q - 2{,}8412).$$

Время T_{O}^{3} сравнения двух штрафов на одном процессоре через время выполнения элементарной операции имеет вид

$$T_{O}^{3} = t_{эл.обр.}\, T_{O}^{3} = t_{эл.пер.} \cdot (2m - j + 1) \sum_{i=1}^{\log_q(2m-j+1)} \frac{1}{q^i}.$$

Аппроксимировав эту функцию по переменной q, получим

$$T_{O}^{3} \approx t_{эл.обр.} \cdot (2m - j + 1)(-0{,}31q^2 + 1{,}2843q - 3{,}8412)$$

Тогда время, затрачиваемое на выполнения j-й итерации метода штрафов, можно оценить величиной

$$T_{Ш}^{j} \approx T_{П}^{1,\,j} + (-45 \cdot 10^{-6} q^2 + 9 \cdot 10^{-4} q - 15 \cdot 10^{-4})(2m - j + 1)^2 + (4 \cdot 10^{-3} q^2 - 0{,}1163q + 0{,}768) \times$$
$$\times (2m - j + 1) - 36 \cdot 10^{-4} \cdot q^2 + 0{,}12 \cdot q + 87 \cdot 10^{-4}] \cdot T_{O}^{1,\,j} + (2m - j + 1)T_{П}^{2} + T_{П}^{3} + T_{O}^{3}.$$

Таким образом, общее время, затраченное на получение опорного плана, составит величину

$$T_{Ш} \approx \sum_{j=1}^{2m} \{ T_{П}^{1,\,j} + (-45 \cdot 10^{-6} q^2 + 9 \cdot 10^{-4} q - 15 \cdot 10^{-4})(2m - j + 1)^2 + (4 \cdot 10^{-3} q^2 - 0{,}1163q +$$
$$+ 0{,}768)(2m - j + 1) - 36 \cdot 10^{-4} \cdot q^2 + 0{,}12 \cdot q + 87 \cdot 10^{-4}] \cdot T_{O}^{1,\,j} + (2m - j + 1)T_{П}^{2} + T_{П}^{3} + T_{O}^{3}\}.$$

Подставив в полученное выражение значение величин, выраженное через элементарные операции, получим

$$T_{Ш} = \sum_{j=1}^{2m} \{ \left[m^2 - (j-1) \times \left(m + \frac{(j-1)}{8} \right) \right] t_{эл.пер.} + [(-45 \cdot 10^{-6} q^2 + 9 \cdot 10^{-4} q - 15 \cdot 10^{-4})(2m - j + 1)^2 +$$
$$+ (4 \cdot 10^{-3} q^2 - 0{,}1163q + 0{,}768)(2m - j + 1) - 36 \cdot 10^{-4} \cdot q^2 + 0{,}12 \cdot q + 87 \cdot 10^{-4}] 2(m - j - 1) \times$$
$$\times t_{эл.обр.} + 3(2m - j + 1)t_{эл.пер.} + 3t_{эл.пер.} \cdot (2m - j + 1)(-23 \cdot 10^{-3} q^2 + 2{,}2843q - 2{,}8412) +$$
$$+ t_{эл.обр.} \cdot (2m - j + 1)(-31 \cdot 10^{-3} q^2 + 1{,}2843q - 3{,}8412)\}.$$

4. Экспериментальная проверка эффективности параллельного алгоритма

Эффективность приведённого в разделе 3 алгоритма проверялась путём выполнения его на кластере следующей конфигурации:

- 4 вычислительных узла (Intel Pentium 4 2,4 ГГц);
- управляющий узел (Intel Pentium 4 2,4 ГГц).

Узлы объединены между собой сетью Infiniband (пропускная способность 4 Гбит/с).

Результаты вычислительных экспериментов по исследованию параллельного алгоритма представлены в таблице 1, которую иллюстрирует рис. 3.

Согласно полученным результатам, ускорение при небольших размерностях задачи значительно меньше 0,5. Это значит, что алгоритм неэффективно работает при небольших размерностях задачи. Однако, для больших размерностей мы получаем значительный выигрыш (до 130%). Таким образом, при увеличении размерности матриц соотношение операций пересылок и полезных операций уменьшается.

Табл. 1. Результаты вычислительных экспериментов по исследованию параллельного алгоритма

Размерность задачи (m*n)	Время выполнения последовательного алгоритма, с	Параллельный алгоритм					
		1 процессор		2 процессора		4 процессора	
		Время, с	Ускорение	Время, с	Ускорение	Время, с	Ускорение
100	0,0435	1,5735	0,0277	0,9320	0,0467	0,2476	0,1757
500	0,2079	2,1591	0,0963	1,7999	0,1155	0,6837	0,3041
1000	0,4849	3,1953	0,1518	2,2136	0,2191	1,4034	0,3455
10000	3,0424	8,8255	0,3447	6,3273	0,4808	3,2364	1,0491
100000	7,5116	14,0951	0,5329	8,5968	0,8738	6,9902	1,2875

На рис. 6 представлены графически зависимости ускорения параллельного алгоритма от число используемых процессоров при различной размерности задачи.

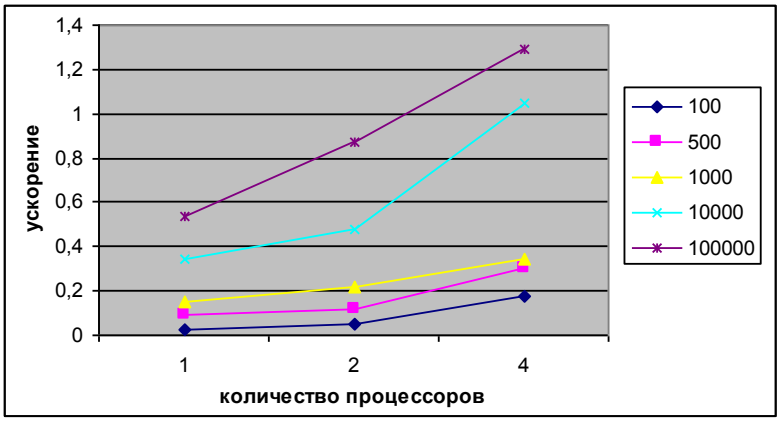


Рис. 6. Зависимость ускорения параллельного алгоритма поиска опорного плана от числа используемых процессоров при различной размерности задач

Сравнение времени выполнения параллельного алгоритма T^*p полученного экспериментальным путем (табл.1) и полученной теоретически оценки времени Tp приведено в табл. 2

Табл. 2. Сравнение экспериментального и теоретического времени работы параллельного алгоритма.

Размерность задачи $(m \cdot n)$	Время выполнения параллельного алгоритма					
	1 процессор		2 процессора		4 процессора	
	T_1, с	T^*_1, с	T_2, с	T^*_2, с	T_4, с	T^*_4, с
100	1,7834	1,5735	0,6710	0,9320	0,2844	0,2476
500	1,8950	2,1591	1,8926	1,7999	0,7561	0,6837
1000	3,0978	3,1953	2,9697	2,2136	1,5135	1,4034
10000	8,9363	8,8255	5,6577	6,3273	3,4711	3,2364
100000	10,1757	14,0951	9,9349	8,5968	7,3644	6,9902

На рис. 7 представлены графики зависимости времени работы разработанного параллельного алгоритма от размерности решаемой задачи, полученные теоретически и экспериментальным путем при использований 4 процессоров

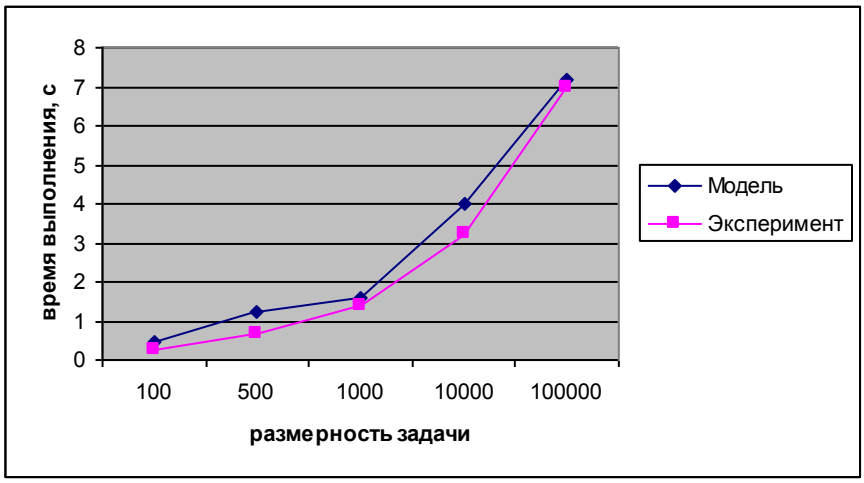


Рис. 7. Графики зависимости времени работы параллельного алгоритма от размерности решаемой задачи.

Из табл. 2 и рис. 7 видно, что полученные теоретические оценки погрешности параллельного алгоритма поиска опорного плана не превышают 15%.

Заключение

Разработан параллельный алгоритм для нахождения опорного плана транспортной задачи, который возможно использовать для работы на кластерной локальной сети, либо на многопроцессорной системе. Проведенное тестирование, позволяет сделать вывод, о том что данный алгоритм возможно использовать для эффективного нахождения опорного плана транспортной задачи.

Опорный план, который получается в результате работы описанного параллельного алгоритма, может быть использован при поиске оптимального решения транспортной задачи. В дальнейшем, возможно разработать параллельный алгоритм для поиска оптимального решения.

Библиографический список

1.Барский А. Б. Параллельное программирование. – СПб.: Бином, 2007.

2.Северин А. А. Исследование эффективности реализации численных методов на кластерах персональных ЭВМ. – Минск, 2005.

3.Хохлюк В. И. Параллельные алгоритмы целочисленной оптимизации. – 2007.

4.Каныгин Г.И., Месхи Б.Ч., Соболь Б.В. Методы оптимизации / Каныгин Г.И., Месхи Б.Ч., Соболь Б.В. — М.: Издательство: Феникс, 2009 г. — 384 с.

5.Рыков А.С. Системный анализ: модели и методы принятия решений и поисковой оптимизации. — М.: МИСиС, 119049.