УДК 519.711.3; 681.51.015; 681.3.01;658.012.011.56:658.512

*Ю.А. Шичкина*

# РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМА НОРМАЛИЗАЦИИ РЕЛЯЦИОННЫХ ОТНОШЕНИЙ

*Одной из важнейших характеристик любой базы данных является ее объем. Современные базы данных охватывают огромное количество атрибутов предметной области, что значительно усложняет математическую модель. Но развитие вычислительных систем с параллельной архитектурой позволяет значительно ускорить процесс обработки этих моделей. В данной статье рассматривается процесс распараллеливания алгоритма нормализации реляционной схемы, с учетом ограничений на время выполнения алгоритма и число процессоров, допускаемых вычислительной системой.*

***Ключевые слова:*** реляционная схема, отношения, нормальная форма, функциональная зависимость, параллельная форма, алгоритм, информационный граф.

Оптимизация информационного графа алгоритма нормализации реляционных отношений по числу процессоров.

С целью разработки оптимального по времени и числу процессоров параллельного алгоритма нормализации реляционного отношения была построена синтаксическая диаграмма. Фактически, весь алгоритм был разбит на сравнительно небольшие операции, и между ними установлены логические и информационные зависимости. Собрав все информационные подграфы блоков синтаксической диаграммы, получаем информационный граф алгоритма нормализации реляционных отношений при $m = 6, n = 11$ (рис. 1).

С помощью алгоритма оптимизации параллельной программы по ширине, без учета времени [2], и программы, созданной в среде Delphi 7.0 и соответствующей этому алгоритму, все вершины графа алгоритма нормализации реляционного отношения были разбиты на группы. Найденные параметры информационного графа алгоритма нормализации реляционных отношений приведены на рис. 2.

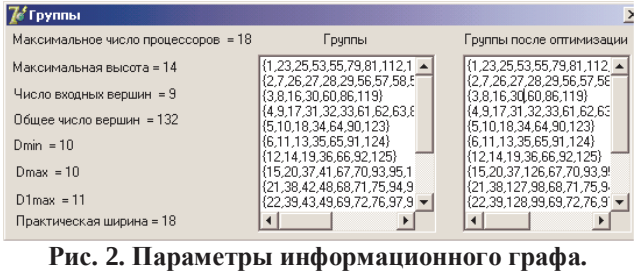



**Рис. 2. Параметры информационного графа.**

С целью уменьшения ширины графа и более равномерного распределения нагрузки на процессоры вычислительной системы было произведено переформирование групп двумя способами:

– только в направлении «вверх», т. е. от последней группы вершин к первой;

– в двух направлениях: вначале «вверх», от последней группы вершин к первой, затем в направлении «вниз», от первой группы вершин к последней. Результаты работы программы по разбиению вершин по группам и оптимизации информационного графа без учета времени двумя способами приведены в таблице 1. Из таблицы 1 видно, что на первых семи этапах работы алгоритма нормализации реляционных отношений равномерно распределить нагрузку между процессорами невозможно, ширина графа колеблется от 7 до 18 процессоров. На последующих шагах некоторые вершины возможно переместить из одной группы в другую. Разность между тремя способами распределения отражает диаграмма на рис. 3. Перемещение вершин в последних группах вторым способом позволяет наиболее равномерно распределить нагрузку между процессорами, настолько, насколько это возможно при данной специфике информационного графа. Разница между максимальной и минимальной шириной графа, к сожалению, сохраняется во всех трех способах распределения вершин.

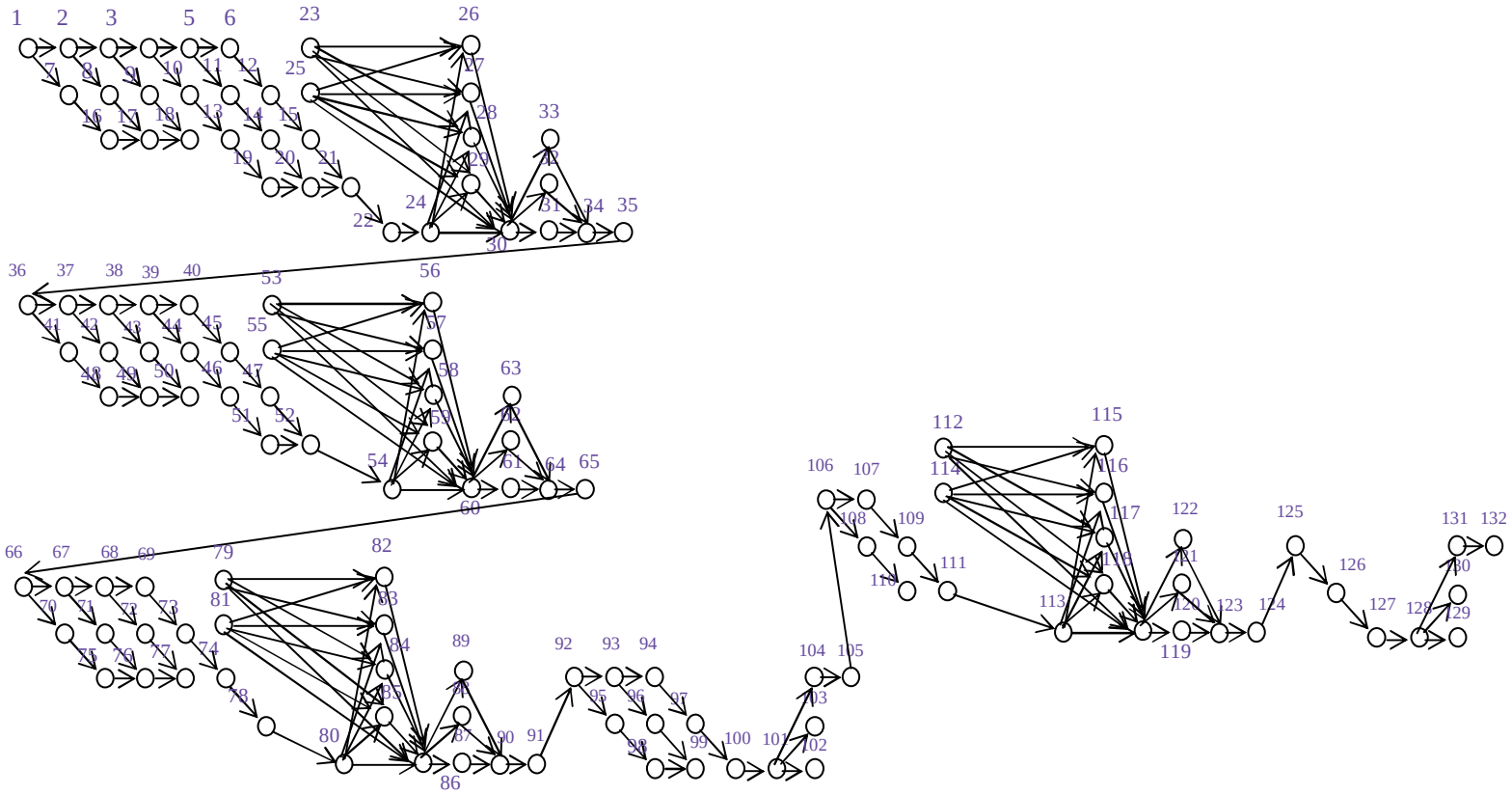


**Рис. 1. Информационный граф алгоритма нормализации реляционных отношений.**

Таблица 1.

*Результаты оптимизации информационного графа по ширине*

| | До оптимизации | После оптимизации «вниз» | После оптимизации «вниз-вверх» |
|---|---|---|---|
| 1. | {1,23,25,53,55,79,81,112,114} | {1,23,25,53,55,79,81,112,114} | {1,23,25,53,55,79,81,112, 114} |
| 2. | {2,7,26,27,28,29,56,57,58,59,82, 83,84,85,115,116,117,118} | {2,7,26,27,28,29,56,57,58,59,82, 83,84,85,115,116,117,118} | {2,7,26,27,28,29,56,57,58,5 9,82,83,84,85,115,116,117, 118} |
| 3. | {3,8,16,30,60,86,119} | {3,8,16,30,60,86,119} | {3,8,16,30,60,86,119} |
| 4. | {4,9,17,31,32,33,61,62,63,87,88, 89,120,121,122} | {4,9,17,31,32,33,61,62,63,87,88, 89,120,121,122} | {4,9,17,31,32,33,61,62,63, 87,88,89,120,121,122} |
| 5. | {5,10,18,34,64,90,123} | {5,10,18,34,64,90,123} | {5,10,18,34,64,90,123} |
| 6. | {6,11,13,35,65,91,124} | {6,11,13,35,65,91,124} | {6,11,13,35,65,91,124} |
| 7. | {12,14,19,36,66,92,125} | {12,14,19,36,66,92,125} | {12,14,19,36,66,92,125} |
| 8. | {15,20,37,41,67,70,93,95,126} | {15,20,37,126,67,70,93,95} | {15,20,37,126,67,70,93,95, 41} |
| 9. | {21,38,42,48,68,71,75,94,96,98, 127} | {21,38,127,98,68,71,75,94,96,41 } | {21,38,127,98,68,71,75,94,9 6,42} |
| 10. | {22,39,43,49,69,72,76,97,99,128 } | {22,39,128,99,69,72,76,97,42,48 } | {22,39,128,99,69,72,76,97, 48,43} |
| 11. | {24,40,44,50,73,77,100,129,130, 131} | {24,40,44,131,73,77,100,130,43, 49} | {24,40,44,131,73,77,100, 130,49,129} |
| 12. | {26,27,28,29,45,46,74,101,132} | {26,27,28,29,45,46,74,101,50, 129} | {26,27,28,29,45,46,74,101, 50,132} |
| 13. | {30,47,51,78,102,103,104} | {30,47,51,78,104,132} | {30,47,51,78,104,102} |
| 14. | {31,32,33,52,80,105} | {31,32,33,52,80,105,102,103} | {31,32,33,52,80,105,103} |

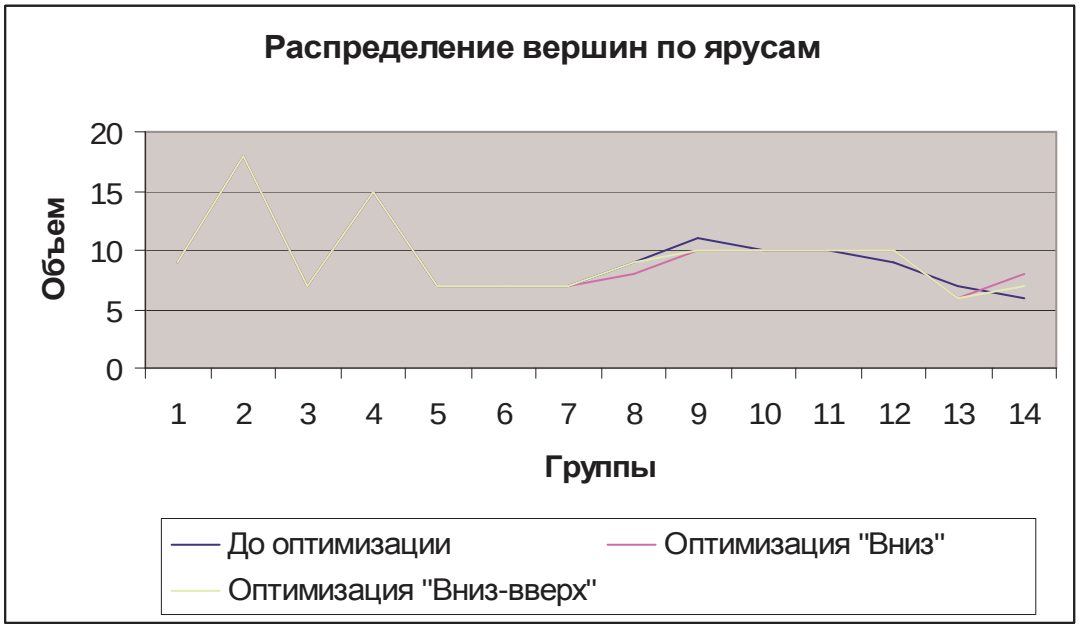



**Рис. 3. Распределение вершин по ярусам графа.**

Тем не менее, проведенная апробация алгоритма оптимизации параллельной программы по ширине еще раз показывает, что последний способ является более эффективным и предлагает разработчику параллельной программы оптимальную

для вычислительной системы с неограниченной степенью параллелизма схему построения параллельной программы.

Оптимизация информационного графа алгоритма нормализации реляционных отношений по времени.

Прежде чем перевести последовательный алгоритм нормализации реляционных отношений в его параллельный аналог, добавим к информационному графу дополнительные характеристики – веса. В качестве веса каждого ребра возьмем примерную временную трудоемкость операции, соответствующей его выходной вершине.

Т. к. ряд операций в алгоритме нормализации реляционных отношений повторяется, в таблице 2 приведен вес только основных операций.

Для расчета времени выполнения операций, приведенных в таблице 2, будем считать, что в данной задаче максимальное число атрибутов в одной строке равно 3. Этот параметр в таблице обозначается $\max\_str$. Также положим максимальное число единиц в строке равным $gk = 6$, максимальное число атрибутов в начале работы алгоритма в строке $r = 9$.

В соответствии с методом расчета ранних сроков выполнения операций информационного графа, время выполнения алгоритма нормализации реляционного отношения при $m = 6, n = 11$ равно 314 ед.

Построенные временные диаграммы для информационного графа до оптимизации и после применения различных алгоритмов оптимизации дают интересные результаты.

Так, временная диаграмма, построенная для параллельной формы информационного графа в соответствии с группами, полученными по информационному графу до применения алгоритма оптимизации по ширине (рис. 3) показывает, что время выполнения алгоритма нормализации реляционного отношения

Таблица 2

*Временная трудоемкость операций*

| № узла | Трудоемкость в общем виде | Численное значение |
|---|---|---|
| 1-6 | $\max\_str$ | 3 |
| 7-15 | $2n/3+1$ | 9 |
| 16-22 | $n/3$ | 4 |
| 23 | $3n/2$ | 17 |
| 24 | $nr/2$ | 50 |
| 25 | $3\max\_str+4$ | 13 |
| 26-30 | $(2m\cdot\max\_str+n)/5$ | 10 |
| 31 | $2r-1$ | 17 |
| 32 | $n/2+1$ | 7 |
| 33 | $3n/2+2$ | 20 |
| 34 | 2 | 2 |
| 35 | $n/2+(m-1)n/2+2$ | 34 |
| 101 | $nr/2$ | 50 |
| 102 | $4\max\_str$ | 12 |
| 103 | $5n/2$ | 26 |
| 104 | $(m-1)n/2+2$ | 28 |
| 105 | $((n-1)m+2)n/3$ | 23 |

может быть сокращено до 168 ед. при 18 процессорах. При этом из диаграммы видно, что все процессоры содержат большие «временные пузыри», т. е. значительное время, в среднем больше половины от всего времени работы алгоритма, процессоры простаивают.

Временная диаграмма, построенная для преобразования в соответствии с группами, полученными после применения алгоритма оптимизации по ширине (рис. 4), показывает, что:

– ширина информационного графа не изменилась и осталась равной 18 процессорам. Таким образом, на практике не всегда удается простым перемещением операций из группы в группу сократить ширину графа;

– высота информационного графа не изменилась и осталась равной 19 ярусам. Это свойство заложено в самом алгоритме;

– время простоев процессоров увеличилось;

– как следствие из предыдущего пункта, время выполнения алгоритма увеличилось до 182 ед.

Таким образом, можно сделать вывод, что в применении к реальной практической задаче алгоритм оптимизации по ширине далеко не всегда приводит к положительному результату.

Применение алгоритма оптимизации информационного графа по времени показало (рис. 5), что время выполнения рассматриваемого параллельного алгоритма можно сократить до 120 ед., сохранив при этом, как и было заложено при разработке алгоритма оптимизации по времени, ширину информационного графа в пределах 18 процессоров. Время простоев процессоров при такой реализации параллельного алгоритма значительно сокращается.

Оптимизация параллельного алгоритма по плотности вычислений.

С целью полного уничтожения «временных пузырей» алгоритм оптимизации информационного графа несколько видоизменили, совместив поэтапно алгоритмы оптимизации по ширине и времени. Результатом симбиоза этих алгоритмов является следующий алгоритм.

Алгоритм оптимизации по плотности вычислений.

1. Выполнить часть 1 алгоритма оптимизации по ширине, т. е. разбить операции на группы.

2. Найти суммарное время работы всех процессоров без простоев, т. е. объем работ V.

3. Применить алгоритм оптимизации по времени.

4. Определить минимальное число процессоров, необходимых для выполнения полученного объема работ V за время Tmin, полученное после оптимизации по времени: $D\min = \frac{V}{T\min}$.

5. Расписать операции по процессорам. Это те же группы, но в данном случае будет удобнее оперировать понятием «процессор». Провести перемещение операций по принципу: количество процессоров не должно превышать минимальную ширину Dmin. Перемещение производить по правилам:

– найти на очередном шаге максимальное время окончания работы процессора;

– перебрать последовательно все последние операции на процессорах, кроме процессора с максимальным временем окончания, и попробовать их переставить на другие процессоры во «временные пузыри»

При этом должны учитываться информационные связи и объемы «пузыря» и переставляемой операции.

Пример. Пусть Dmin=3. После применения алгоритма оптимизации по времени получена временная диаграмма:

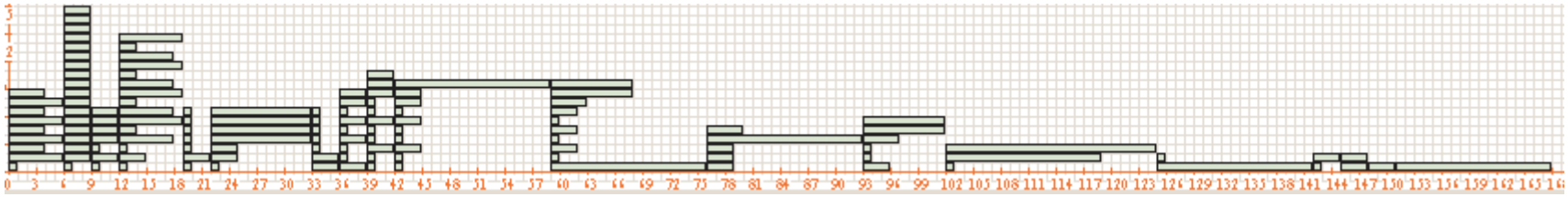


**Рис. 3. Временная диаграмма до оптимизации по ширине.**

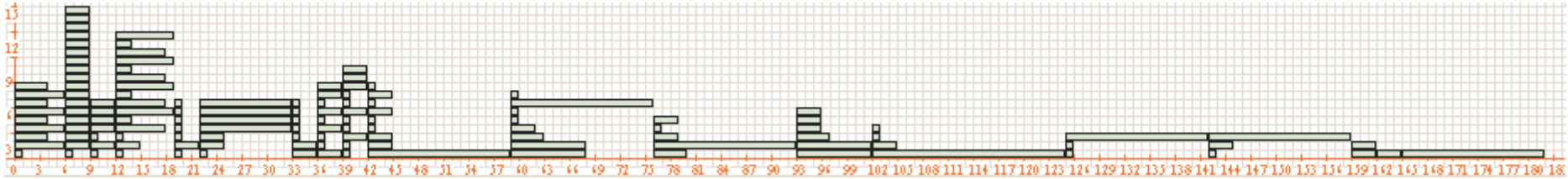


**Рис. 4. Временная диаграмма после оптимизации по ширине.**

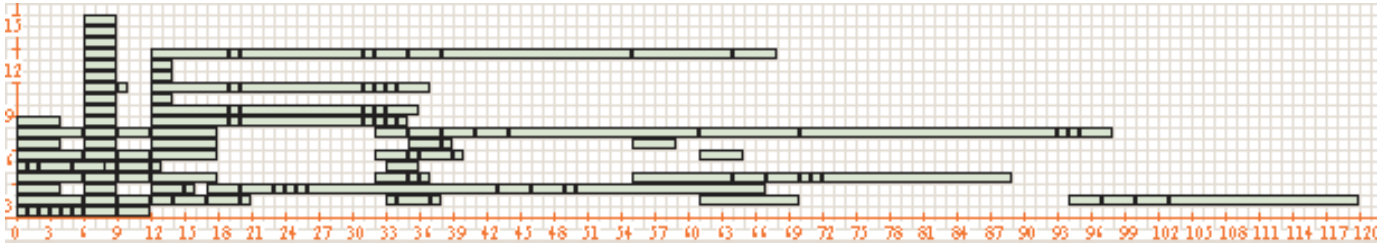


**Рис. 5. Временная диаграмма после оптимизации по времени.**

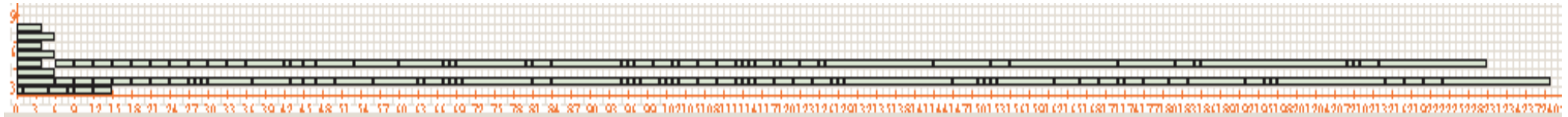
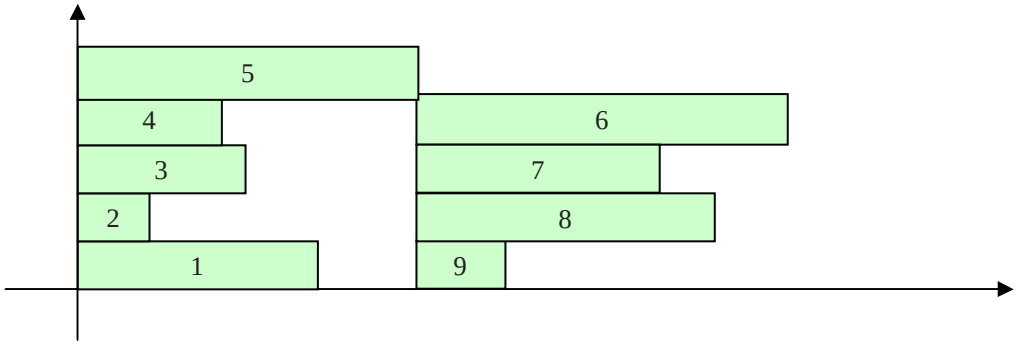
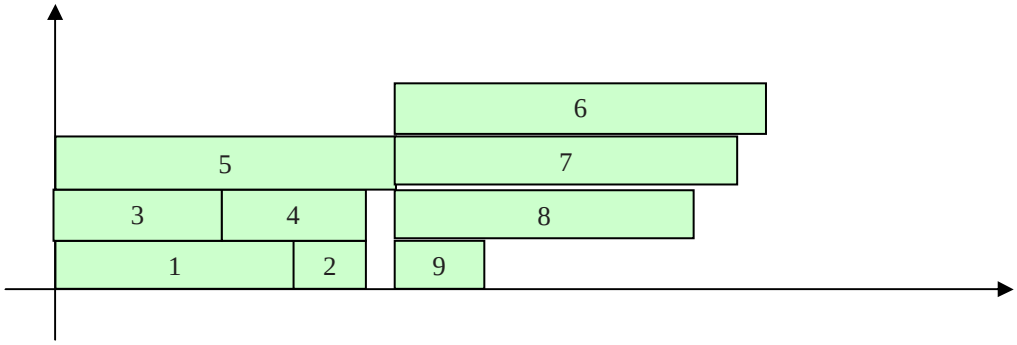


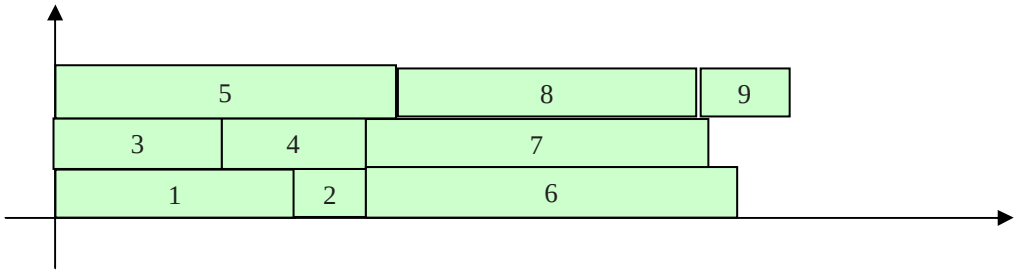
**Рис. 6. Временная диаграмма после оптимизации по плотности вычислений.**



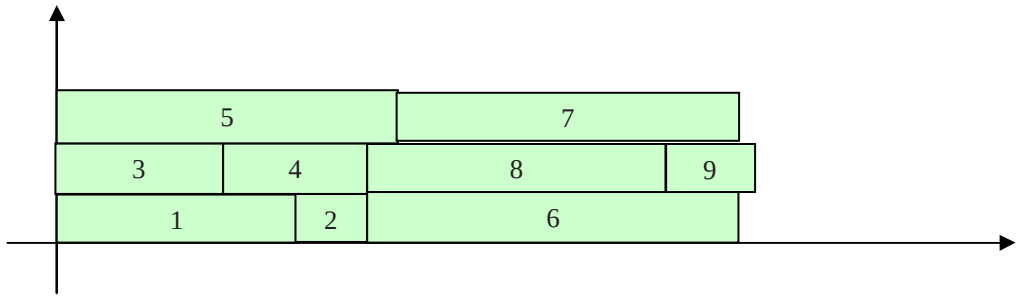

На первом шаге перемещаем операции 2 и 3.




На втором шаге можно переместить операции таким образом:




или



По данному алгоритму будет выбран последний вариант, т. к. он экономичнее по времени.

Несмотря на экономию по времени за счет того, что многие операции невозможно будет из-за информационной зависимости переставить на определенный процессор, т. к. произойдет обрыв работы, все остальные операции лягут на оставшиеся процессоры, и общее время выполнения алгоритма будет увеличено.

Недостатком полученного алгоритма является частое увеличение времени выполнения всего алгоритма.

Результаты применения алгоритма оптимизации информационного графа по плотности вычислений приведены на рис. 6. В применении к задаче распараллеливания процесса нормализации реляционного отношения алгоритм оптимизации информационного графа по плотности вычислений позволяет:

– сократить число процессоров до 8;

– полностью убрать простои процессоров. Уже после 15 ед. времени в работе остаются всего 2 процессора, остальные полностью освобождаются от решения данной задачи и могут быть использованы в работе над другими задачами;

– увеличить время выполнения всего алгоритма до 241 ед., что лучше по сравнению с алгоритмом расчета ранних временных сроков, но хуже в сравнении с результатом работы алгоритма оптимизации информационного графа по времени.

Заключение

На основании проведенной апробации различных алгоритмов оптимизации информационного графа можно сформулировать следующие выводы.

1. Оптимизацию информационного графа на основе матрицы смежности можно проводить по трем направлениям: по ширине, высоте и плотности вычислений.

2. При подборе алгоритмов оптимизации следует учитывать два параметра – время и число доступных процессоров.

3. В зависимости от информационного графа, все алгоритмы могут давать различные по качеству результаты. Следовательно, имея в наличии информационный граф и его матрицу смежности, желательно применить все алгоритмы оптимизации и на основе сравнительного анализа полученных результатов выбрать оптимальный способ решения задачи распараллеливания последовательного алгоритма.

4. Экономия времени при выполнении параллельной программы, по сравнению с ее последовательным аналогом, с учетом времени, затраченного на построение информационного графа, и выбора способа распараллеливания зависит от объема данных и структуры последовательного алгоритма.

В целом, каждый из полученных алгоритмов дает в той или иной степени положительный результат.

*Литература*

1. Шичкина Ю. А. Автоматизация процесса нормализации реляционных баз данных // Наука. Технологии. Инновации : материалы всерос. науч. конф. молодых ученых : в 6 ч. Новосибирск : НГТУ, 2004. Ч 1. С. 181-187.

2. Шичкина Ю. А., Воробьев В. И. Оптимизация параллельного алгоритма по числу процессов. Вестн. гражданских инженеров. 2008. № 2 (15). С. 92 - 97.

3. 3. Шичкина Ю. А. Сокращение высоты информационного графа параллельного алгоритма // Научн.-техн. ведомости СПбГПУ. Сер. Информатика. Телекоммуникации. Управление. 2009. № 3 (80). С. 148 - 152.