М.С.Куприянов
Ю.А.Шичкина

Метод эквивалентного преобразования параллельных алгоритмов для обеспечения их эффективной реализации в распределенных системах

Аннотация

Не эффективно работающая программа – это прямые потери производительности компьютера. Таких потерь хотелось бы избежать или, по крайней мере, их минимизировать. Для этого необходимы апробированные методы исследования, разработки и эквивалентных преобразований алгоритмов. В данной работе предлагается метод модификации параллельного алгоритма с целью повышения его эффективности использования за счет обеспечения равномерной загрузки вычислительных ядер. Метод заключается в перестановке процессов между ядрами и укрупнения операций алгоритма. Все информационные зависимости алгоритма при этом сохраняются, время выполнения алгоритма и количество задействованных процессоров может быть только сокращено.

Введение

При создании параллельных вычислительных технологий возникает необходимость в создании методов распараллеливания алгоритмов и программ, а также в методах, обеспечивающих эффективное распределение процессов между вычислительными ядрами В последнее время параллельным вычислениям было посвящено много работ. В целом все эти работы можно было бы условно разделить на три основные категории:

– Работы в области теории параллельного программирования [6, 7, 16] с описанием в качестве примеров достаточно сложных задач и алгоритмов, например, многосеточных методов [4, 5] и др.

– Работы, посвященные построению параллельных алгоритмов для задач узкого класса из некоторой области с примерами параллельных алгоритмов вычислительной математики [8, 14] и др.

– Работы зарубежных и отечественных ученых, в которых были предложены формальные модели, позволяющие описывать функционирование последовательных процессов, исполняющихся параллельно[1-3, 6, 7, 9] и др.

Широкое освещение результатов решения проблемы планирования вычислительного процесса можно найти в работах [11-13, 15] и др.

Чаще всего оптимизация параллельных алгоритмов рассматривается в условиях неограниченного параллелизма, которые предполагают использование идеализированной модели параллельной вычислительной системы. Условия неограниченного параллелизма – это идеальные условия, которые на практике не существуют. В связи с этим, хотелось бы найти новые методы модификации параллельных алгоритмов, которые были бы более приближены к реальным условиям.

Эффективность параллельного алгоритма зависит от многих параметров. Одним из таких параметров является равномерная загрузка процессоров вычислительной системы. При этом важно с одной стороны сохранив высоту информационного графа параллельного алгоритма минимизировать его ширину. Другим важным параметром является время выполнения процессов. Если предполагать, что время выполнения всех процессов, отраженных в информационном графе в виде отдельных вершин, одинаково, тогда ориентированный ациклический информационный граф является наиболее подходящим инструментом для построения оптимального по высоте и ширине параллельного алгоритма. Но на практике такие условия – крайняя редкость. Следовательно, из-за неравномерности выполнения отдельных процессов, часть процессоров будет простаивать в ожидании результатов от своих предшественников. Поэтому, сама собой напрашивается замена ориентированного ациклического информационного графа его взвешенным аналогом, в котором в качестве весов будет выступать время выполнения процессов.

При исследовании информационного графа возникают следующие вопросы:

- ✓ Насколько непрерывно работают все три задействованные в вычислительном процессе системе? Сколько времени простаивает каждый процессор в процессе работы алгоритма?
- ✓ Можно ли сократить общее время работы алгоритмы путем уменьшения времени простоев процессоров?
- ✓ Можно ли сократить ширину алгоритму путем уменьшения времени простоев процессоров за счет объединения мелких (по времени работы) операций в более крупные?

Метод поиска оптимального по равномерности использования вычислительных ядер информационного графа

Предлагаемый метод оптимизации информационного графа основан на списках следования и заключается в разбиении совокупности вершин на группы (построение параллельной формы) и перемещение вершин между группами с целью уменьшения числа ярусов и достижения максимальной плотности по каждому ярусу. Суть метода состоит в следующем:

1. Рассчитать значение теоретической минимальной ширины d информационного графа.
2. Добавить в граф вершину. Все выходные вершины соединить с новой вершиной. Таким образом, в графе будет всего одна выходная *n+1*-я вершина.

3. Составить матрицу следования информационного графа и ее замыкания A. Вместо единиц в матрице следования будут стоять значения времени выполнения j-й операции.

4. Построить соответствующий информационному графу временной список следования $V_0 = (V_1, V_2, t)$.

5. Построить множество выходных вершин V_k.

6. Найти множество $V = V_2 - V_1$. Вершины, вошедшие в это множество, составят группу вершин M_i, принадлежащих одному i-му ярусу.

– Если число вершин множества $V = V_2 - V_1$ превышает d, то возможны следующие варианты:

a) во множестве M_i существует ровно d вершин с одинаковым временем выполнения. В этом случае эти d вершин с одинаковым временем выполнения составят множество M_i.

b) Во множестве M_i существует больше чем d вершин с одинаковым временем выполнения. В этом случае любые d вершин с одинаковым временем выполнения составят множество M_i.

c) Во множестве M_i существует меньше d вершин с одинаковым временем выполнения. В этом случае множество M_i составят d вершин с минимальным временем выполнения.

Остальные вершины необходимо включить в следующую группу M_{i+1}. При повторном проходе этого шага в группу M_{i+1} будут добавлены вершины

– Если число вершин множества $V = V_2 - V_1$ меньше числа d, то во множество M_i следует перенести вершины из множества $V_к$, удовлетворяющие правилу: $V_{kj} \cap V_1 = \varnothing$, т.е. вершины из $V_к$, отсутствующие во множестве V_1. Эти вершины могут быть взяты в произвольном порядке из множества: $V_k - V_1$.

7. Удалить из списка смежности все пары, конечная вершина которых (V_2), совпадает с одной из вершин множества V и построить тем самым список V_i.

8. Если i=1, то вернуться на шаг 6.

9. Во множестве M_{i-1} найти вершину с минимальным временем выполнения m_{i-1k}. Если все вершины имеют одинаковое время выполнения, то перейти к шагу 16.

10. Составить множества:

- $S_{m_{i-1k}M_i}$ – совокупность вершин из множества M_i, связанных с вершиной m_{i-1k} одним ребром и являющихся для данного ребра конечными вершинами.
- $S_{(M_{i-1}-m_{i-1k})M_i}$ – совокупность вершин из множества M_i, связанных с вершинами множества M_{i-1} (за исключением вершины m_{i-1k}) одним ребром и являющихся для данного ребра конечными вершинами.

  Внимание! Если во множестве M_{i-1} есть укрупненные операции, то рассматривается всегда только последняя добавленная при укрупнении вершина.

11. Найти разность: $S = S_{m_{i-1k}M_i} - S_{(M_{i-1}-m_{i-1k})M_i}$.

- Если $S = \varnothing, S_{m_{i-1k}M_i} \neq \varnothing, S_{(M_{i-1}-m_{i-1k})M_i} \neq \varnothing$, то перейти к шагу 5.
- Если $S_{m_{i-1k}M_i} = \varnothing, S_{(M_{i-1}-m_{i-1k})M_i} = \varnothing$, то $S = M_i$.
- Если $S_{m_{i-1k}M_i} = \varnothing, S_{(M_{i-1}-m_{i-1k})M_i} \neq \varnothing$, то $S = M_i - S_{(M_{i-1}-m_{i-1k})M_i}$.

12. Найти во множестве вершину с минимальным временем выполнения $s_{\min}$. Удалить эту вершину из множества M_i. Добавить эту вершину во множество M_{i-1}, укрупнив операцию m_{i-1k} путем ее слияния с операцией $s_{\min}$:

$$m_{i-1k} = m_{i-1k} \cup s_{\min}, t_{i-1k} = t_{i-1k} + t_{s_{\min}}$$

13. Если $M_i = \varnothing$, то удалить множество $M_i = \varnothing$ и сдвинуть счетчик множеств $i = i - 1$. Перейти к шагу 5. Если $M_i \neq \varnothing$, то перейти к шагу 10.

14. Если список не пустой, то вернуться на шаг 5.

15. Если в списке не осталось ни одной пары, то метод окончен.

Общее время работы алгоритма составит $T = \sum_{i=1}^{k} t_{i,\max}$, где k – число групп, $t_{i,\max}$ – максимальное время среди операций в i-й группе.

Пример работы метода

Применим полученный метод к информационному графу (рис 1).

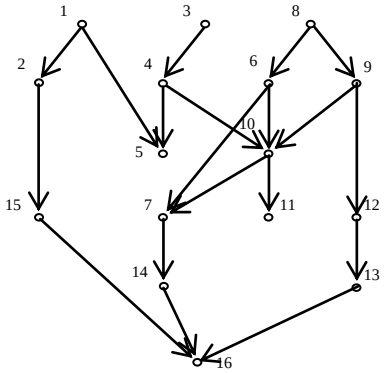


Рисунок 1. – Тестовый граф.

Время выполнения операций составляет: $\{2,1,3,2,2,1,2,3,2,1,3,3,1,2,3,1\}$.

Составим для заданного информационного графа список следования V_0, начало которого будет выглядеть следующим образом:

V_1	V_2	t
6	8	3
9	8	3
4	3	3
12	8	3
12	9	2
…	…	…

Найдем разность множеств $V = V_2 - V_1 = \{1, 3, 8\}$. Эта группа вершин составит первый ярус информационного графа: $M_1 = \{1, 3, 8\}$:

M_1	
i	t
1	2
3	3
8	3

Теоретически минимальная ширина графа равна 3. Поэтому в эту группу пока не будем добавлять вершины из конечной группы V_k. Удалим из первоначального списка V_0 все пары, вторым элементом которых является одна из вершин группы M_1 и получим новый список V_1.

Вновь найдем разность множеств конечных и начальных вершин ребер графа $V = V_2 - V_1 = \{9, 4, 6, 2\}$.

M_2	
j	t
9	2
4	2
6	1

2	1

Во множестве M_1 найдем вершину с минимальным временем выполнения $m_{11}=1, t_1=2$.

Составим множества S_{1M_2} (совокупность вершин из множества M_2, связанных с вершиной 1 одним ребром и являющихся для данного ребра конечными вершинами) и $S_{\{3,8\}M_2}$ (совокупность вершин из множества M_2, связанных с вершинами множества M_1 за исключением вершины 1 одним ребром и являющихся для данного ребра конечными вершинами). Найдем разность: $S = S_{1M_2} - S_{\{3,8\}M_2}$.

S_{1M_2}	$S_{\{3,8\}M_2}$	$S = S_{1M_2} - S_{\{3,8\}M_2}$
4	4	2
6	9	
2	6	

Во множестве S одна вершина 2. Удалим эту вершину из множества M_2 и добавим ее во множество M_1, укрупнив тем самым операцию 1 путем ее слияния с операцией 2: *1'={1,2}, $t_1=t_1+t_2=2+1=3$.*

Скорректированные группы				
M_1			M_2	
i	t		j	t
1,2	3		9	2
3	3		4	2
8	3		6	1

В результате, первая группа выровнена по времени. На этом ее фиксируем. Дальнейшим изменениям данная группа не подлежит.

Перейдем к составлению третьей группы. Удалим из списка V_1 все пары, первым элементом которых является одна из вершин группы M_2 и получим новый список V_2.

В списке V_2 найдем разность множеств конечных и начальных вершин ребер графа $V = V_2 - V_1 = \{10, 12, 15, 5\}$.

M_3	
k	*t*
10	1
12	3
15	3
5	2

Во множестве M_2 найдем вершину с минимальным временем выполнения $m_{23}=6, t_6=1.$

Составим множества S_{6M_3} (совокупность вершин из множества M_3, связанных с вершиной 6 одним ребром и являющихся для данного ребра конечными вершинами) и $S_{\{9,4\}M_3}$ (совокупность вершин из множества M_3, связанных с вершинами множества M_2 за исключением вершины 6 одним ребром и являющихся для данного ребра конечными вершинами). Найдем разность: $S = S_{6M_3} - S_{\{9,4\}M_3}$.

S_{6M_3}	$S_{\{9,4\}M_3}$	$S = S_{6M_3} - S_{\{9,4\}M_3}$
10	10	15
15	12	
	5	

Во множестве S одна вершина 15. Удалим эту вершину из множества M_3 и добавим ее во множество M_2, укрупнив тем самым операцию 6 путем ее слияния с операцией 15: *6'={6,15}, $t_6=t_6+t_{15}=1+3=4$.*

Скорректированные группы			
M_2		M_3	
j	t	k	t
9	2	10	1
4	2	12	3
6,15	4	5	2

В результате, во второй группе появились две вершины со временем выполнения меньше общего группового времени. Группа не выровнена по времени. Повторяем последние два шага для вершин с минимальным временем.

Во множестве M_2 найдем вершину с минимальным временем выполнения: $m_{21}=9, m_{22}=4, t_9=t_4=2$. Выделим любую из них, например, 9.

Составим множества S_{9M_3} и $S_{\{4,15\}M_3}$.

Внимание. Вершину 6 не рассматриваем, т.к. она перекрыта вершиной 15. Найдем разность: $S = S_{9M_3} - S_{\{4,15\}M_3}$.

S_{9M_3}	$S_{\{4,15\}M_3}$	$S = S_{9M_3} - S_{\{4,15\}M_3}$
10	10	12
12	5	

Во множестве S одна вершина 12. Удалим эту вершину из множества M_3 и добавим ее во множество M_2, укрупнив тем самым операцию 9 путем ее слияния с операцией 12: *9'={9,12}*, $t_9=t_9+t_{12}=2+3=5$.

Скорректированные группы			
M_2		M_3	
j	t	k	t
9,12	5	10	1
4	2	5	2
6,15	4		

Вторая группа не выровнена по времени, а в третьей группе есть еще вершины. Повторяем последние два шага для вершин с минимальным временем.

Во множестве M_2 найдем вершину с минимальным временем выполнения $m_{22}=4, t_4=2$.

Составим множества S_{4M_3} и $S_{\{12,15\}M_3}$. Найдем разность: $S=S_{4M_3}-S_{\{12,15\}M_3}$.

S_{4M_3}	$S_{\{12,15\}M_3}$	$S=S_{4M_3}-S_{\{12,15\}M_3}$
5	$\varnothing$	5
10		10

Во множестве S две вершины 5 и 10. Удалим из множества M_3 вершину с минимальным временем выполнения и добавим ее во множество M_2, укрупнив тем самым операцию 4 путем ее слияния с операцией 10: *4'={4,10}*, $t_4=t_4+t_{10}=2+1=3$.

Скорректированные группы			
M_2		M_3	
j	t	k	t
9,12	5	5	2
4,10	3		
6,15	4		

Вторая группа не выровнена по времени, а в третьей группе есть еще вершины. Повторяем последние два шага для вершин с минимальным временем.

Во множестве M_2 найдем вершину с минимальным временем выполнения: $m_{22}=\{4,10\}==10,\ t_{10}=3$.

Составим множества S_{10M_3} и $S_{\{12,15\}M_3}$.

Внимание. Вершину 6 не рассматриваем, т.к. она перекрыта вершиной 15. Найдем разность: $S=S_{10M_3}-S_{\{12,15\}M_3}$.

S_{10M_3}	$S_{\{12,15\}M_3}$	$S = S_{10M_3} - S_{\{12,15\}M_3}$
∅	∅	5

Так как, оба множества S_{10M_3} и $S_{\{12,15\}M_3}$ являются пустыми, то их разность тоже будет пустой. В этом случае любая вершина из множества M_3 может быть перенесена во множество M_2. Поэтому в графе $S = S_{10M_3} - S_{\{12,15\}M_3}$ отображена единственная вершина множества M_3, т.е. 5. Удалим эту вершину из множества M_3, и добавим ее во множество M_2, укрупнив тем самым операцию {4,10} путем ее слияния с операцией 5: *{4,10}'={4,10,5}, $t_{4,10}=t_{4,10}+t_5=3+2=5$.*

Скорректированные группы			
M_2		M_3	
j	t	k	t
9,12	5	∅	
4,10,5	5		
6,15	4		

Вторая группа не выровнена по времени, но в третьей группе больше вершин нет. Удаляем третью группу. Считаем i=2 и переходим к созданию новой группы из списка V_2. Алгоритм продолжается до тех пор, пока разность множеств $V_1 V_2$ не станет пустой:

№	$V_1 V_2$	t
	∅	

В результате мы получим три группы:

M_1	
i	t
1,2	3
3	3
8	3

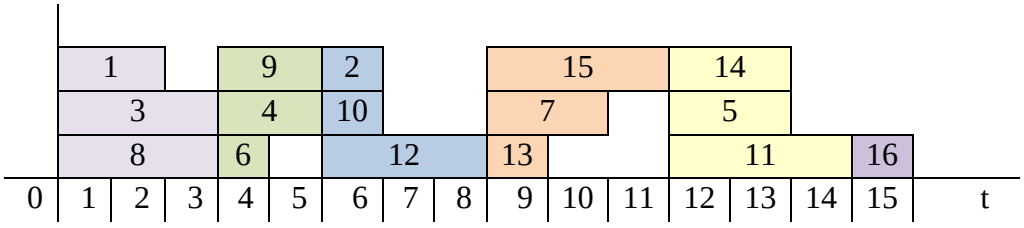
M_2	
j	t
9,12,13,14	8
4,10,5,11	8
6,15,7	6

M_3	
b	t
16	1

Общее время работы алгоритма составит $T = \sum_{i=1}^{k} t_{i,\max} = 3+8+1=12$.

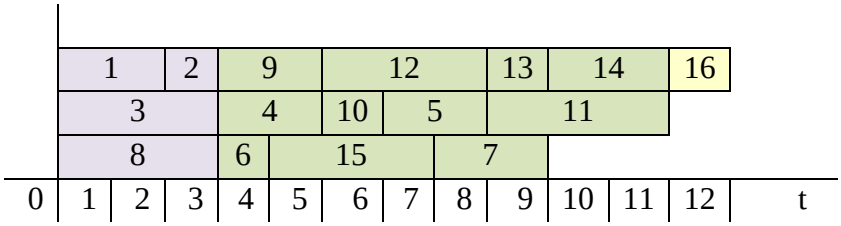
Проиллюстрируем результаты с помощью временных диаграмм.

До применения метода оптимизации по времени на основе любого из методов оптимизации по ширине были получены шесть групп операций для трех процессоров:



Время работы алгоритма равно 15 ед.

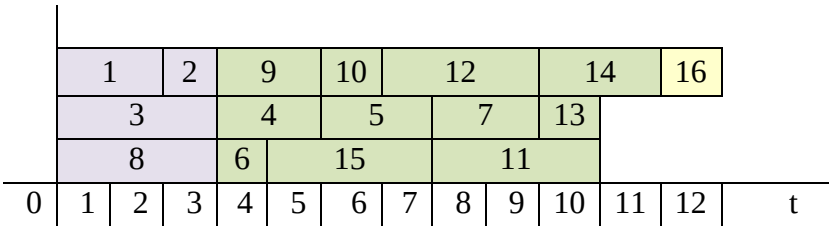
После применения метода оптимизации по времени с сохранением минимальной теоретической ширины d=3 было получено три группы:



Время работы алгоритма равно 12 ед.

Анализ трудоемкости показал, что применение временных списков следования является более эффективным по сравнению с аналогичным методом, состоящим из двух частей (п.3.3.1) и основанном на матрице смежности. Метод оптимизации информационного графа по времени с применением временных списков следования позволяет значительно сократить время работы алгоритма с сохранением заданной ширины графа. При этом ширина графа может быть оговорена произвольным образом: либо на основании найденной теоретической минимальной ширины графа, либо в соответствии с особенностями вычислительной системы.

Следует заметить, что решение задачи минимизации алгоритма по времени вычисления может быть не единственным. Применение вышеизложенного метода оптимизации информационного графа по времени с применением временных списков следования позволяет свести время выполнения алгоритма к минимальному при заданной ширине, и найти только одно из решений задачи минимизации. Так, в приведенном примере, существует еще одно решение:



Время работы алгоритма равно 12 ед.

Следует отметить, что помимо уменьшения времени выполнения параллельного алгоритма данный метод укрупнения схемы позволяет в некоторых случаях сократить и ширину алгоритма за счет укрупнения операций и образования пустот (простоев), которые можно заполнить операциями с других процессов.

Тестирование метода

Для тестирования приведенного метода оптимизации параллельных программ по загрузке вычислительной техники было создано программное обеспечение, позволяющее:

- ✓ строить автоматически ориентированные графы, соответствующие информационным графам в строгой параллельной форме;

✓ строить автоматически взвешенные ориентированные графы, соответствующие информационным графам в строгой параллельной форме (рисунке 2);

✓ строить вручную произвольные ориентированные графы и задавать веса;

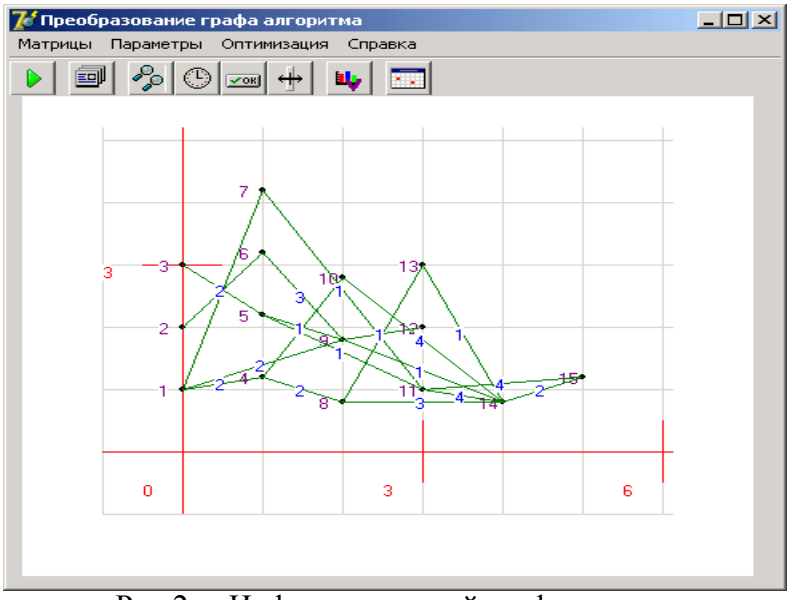


Рис 2. – Информационный граф алгоритма.

✓ рассчитывать высоту, ширину, теоретическую и практическую минимальную ширину графа, ранние и поздние строки выполнения операций и др. параметры (рис. 3, 4).

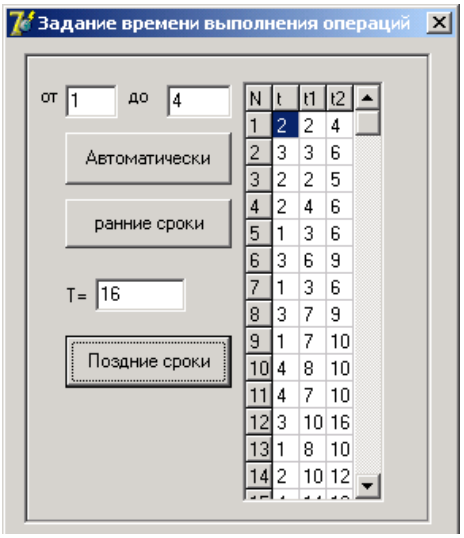


Рисунок 3. – Расчет ранних и поздних сроков выполнения операций

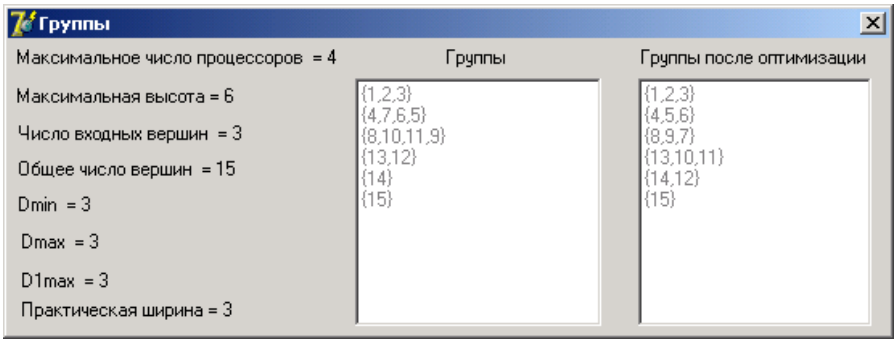


Рисунок 4. – Расчет дополнительных параметров

✓ оптимизировать информационный граф по ширине (рисунок 4);

✓ оптимизировать информационный граф по времени выполнения;

✓ оптимизировать информационный граф по ширине и времени выполнения комбинированным методом, полученным из объединения и модификации методов оптимизации.

✓ строить временные диаграммы (рис. 5):

На рисунке 5 приведена временная диаграмма исходного информационного графа (рисунка 2). Время выполнения алгоритма в соответствии с этим информационным графом равно 18 ед., число процессоров – 4. Последующие три диаграммы иллюстрируют работу методов оптимизации: по ширине (процессоров стало 3), по времени (время сократилось до 15 ед.).

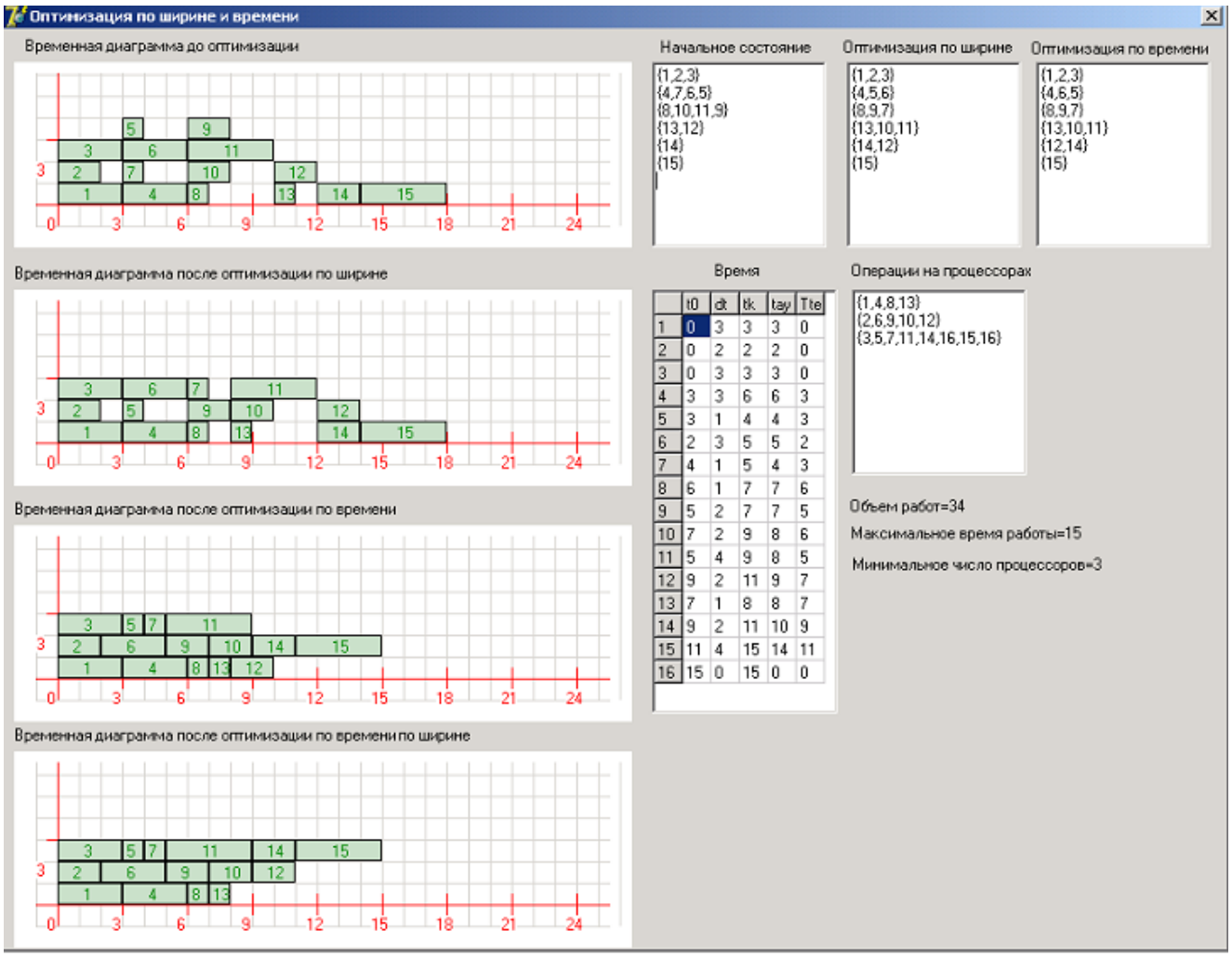


Рисунок 5 – Построение временных диаграмм.

- ✓ построение матриц смежности, следования, списков связности;
- ✓ вывод совокупностей операций, выполняемых каждым процессором.

Выводы

Разработанный метод оптимизации информационного графа параллельного алгоритма, основанный на списках следования обеспечивает равномерное заполнение групп операций, выполняемых параллельно и построен таким образом, что ширина каждого яруса будет приближена к теоретической минимальной ширине, никогда не превосходя ее. Метод является итерационным и решение считается найденным в случае, когда полученное решение i-го шага равняется решению i-1-шага. Модель параллельного

алгоритма, получаемая в результате применения данного метода, может служить образцом при физической реализации параллельного алгоритма.

Разработанное программное обеспечение для тестирования полученного метода и проведения дальнейших исследований в данной области позволят разными способами строить информационный граф (с помощью инструментов рисования, ввода матрицы смежности, загрузки матрицы смежности из файла Excel, автоматической генерации графа), строить модель параллельного алгоритма без оптимизации и с оптимизацией по числу процессоров, времени выполнения, плотности вычислений или по нескольким параметрам одновременно, строить временные диаграммы и масштабировать их, проводить расчет количественных характеристик исследуемого графа (различные оценки ширины и высоты графа, ранние и поздние сроки выполнения операций, загрузку процессоров). Программа разработана в среде Microsoft Visual Studio 2010 на языке C#, зарегистрирована в реестре Федеральной службы по интеллектуальной собственности, патентам и товарным знакам 10.11.2012. свид. ГР. №2012618117.

Литература

1. Ahmad, I. A Parallel Algorithm for Optimal Task Assignment in Distributed Systems / I. Ahmad, M. Kafil // Proceedings of the 1997 Advances in Parallel and Distributed Computing Conference. – 1997. – P. 284.
2. Chen, Hu. MPIPP: An Automatic Profileguided Parallel Process Placement Toolset for SMP Clusters and Multiclusters / Hu.Chen // Proceedings of the 20th annual international conference on Super-computing. – 2006. – P. 353 – 360.
3. Rauber, N., Runger, G. Parallel Programming: for Multicore and Cluster Systems. / N. Rauber, G. Runger. – Springer, 2010. – 450p.
4. Богачев, К. Ю. Основы параллельного программирования / К.Ю. Богачев – М: Бином, 2010. – 344 с.
5. Богданов, А.В., Корхов, В.В., Мареев, В.В., Станкова, Е.Н. Архитектуры и топологии многопроцессорных вычислительных систем / А.В.Богданов, В.В.Корхов, В.В.Мареев, Е.Н.Станкова – М.: Изд-во «Интернет-университет информационных технологий – Интуит.ру», 2004. – 176с.
6. Воеводин, В.В., Воеводин, Вл.В. Параллельные вычисления / В.В.Воеводин, Вл.В.Воеводин –СПб.:БХВ-Петербург, 2004. – 608с.
7. Воеводин, Вл. В., Жуматий, С. А. Вычислительное дело и кластерные системы / Вл.В. Воеводин, С.А. Жуматий – М.: Изд-во МГУ, 2007. – 150с.

8. Гергель, В.П. Теория и практика параллельных вычислений / В.П.Гергель - М.: Интернет-университет информационных технологий.; БИНОМ: Лаборатория знаний, 2007 – 423с.

9. Корнеев, В.Д. Параллельное программирование в MPI / В.Д.Корнеев – Новосибирск: Издательство СО РАН, 2000. – 220с.

10. Корячко, В. П. Иерархическая модель глобальной оптимизации у параллельных объектных программ / В.П.Корячко, С.В. Скворцов // Наука и образование, 2006. – №8. – С.156-164.

11. Костенко, В.А., Романов, В.Г., Смелянский, Р.Л. Алгоритмы минимизации аппаратных ресурсов ВС / Костенко В.А., Романов В.Г., Смелянский Р.Л. - М.: Изд-во МГУ, 2000. – 86с.

12. Коффман, Э.Г. Теория расписаний и вычислительные машины/ Под редакцией Э.Г. Коффмана. -М.: Наука, 1984. -335с.

13. Левин, В.И. Структурно-логические методы в теории расписаний / В.И.Левин – Пенза: Изд-во Пенз. гос. технол. акад., 2006. – 128с.

14. Миллер, Р., Боксер, Л. Последовательные и параллельные алгоритмы с описанием параллельных алгоритмов на графах, умножения матриц, обработки изображений, вычислением многочленов / Р.Миллер, Л.Боксер – Изд-во.: Бином.Лаборатория знаний., 2006. – 408с.

15. Топорков, В.В. Модели распределенных вычислений / В.В.Топорков — М.: Физматлит, 2004. — 315с.

16. Эндрюс, Г.Р. Основы многопоточного, параллельного и распределенного программирования.: пер.с англ. / Г.Р.Эндрюс – М.:Издательский дом «Вильямс», 2003. – 512с.