УДК 519.711.3, 681.51.015, 681.3.01, 658.012.011.56:658.512

Ю. А. Шичкина

КОМБИНИРОВАННЫЙ МЕТОД МНОГОПАРАМЕТРИЧЕСКОЙ ОПТИМИЗАЦИИ ВЗВЕШЕННОГО ИНФОРМАЦИОННОГО ГРАФА

Параллельные программы можно рассматривать как распределенные последовательные взаимодействующие процессы, где проблема разбиения программы на процессы ложится на разработчика. В данной статье рассмотрены аспекты оптимизации параллельного алгоритма с помощью взвешенного информационного графа по двум параметрам: числу процессоров и времени выполнения. Основой алгоритма является перераспределение процессов по процессорам и укрупнение процессов с помощью их объединения.

Ключевые слова: информационный граф, программа, процесс, время выполнения, оптимизация, высота графа.

Введение. До сих пор оптимизация параллельных алгоритмов рассматривалась в условиях неограниченного параллелизма, который предполагает использование идеализированной модели параллельной вычислительной системы. Условия неограниченного параллелизма это идеальные условия, которые на практике не существуют. В связи с этим хотелось бы на основе полученных алгоритмов найти новые, которые были бы приближены к реальным условиям.

Эффективность параллельного алгоритма зависит от многих параметров. Один из них – равномерная загрузка процессоров вычислительной системы. При этом важно, с одной стороны, сохранив высоту 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 алгоритма путем уменьшения времени простоя процессоров?

✓ Можно ли сократить ширину алгоритма путем уменьшения времени простоев процессоров за счет объединения мелких (по времени работы) операций в более крупные?

Алгоритм оптимизации информационного графа параллельного алгоритма. Как показали исследования, модифицированный алгоритм Дейкстры возможно применить и для оптимизации взвешенного графа.

Алгоритм укрупнения схемы является продолжением алгоритма оптимизации ширины [2] и заключается в следующем.

1) Разбить вершины графа на группы с помощью алгоритма оптимизации ширины информационного графа.

2) Начиная с первого яруса, найти группу вершин с разными временными значениями. Отметить ее.

3) В найденной группе найти максимальное значение по времени.

4) Отметить вершину, временное значение которой меньше максимального, и рассмотреть возможность ее объединения с вершиной из следующей группы по принципу:

Если в строке матрицы смежности, соответствующей отмеченной вершине, есть ненулевой элемент, соответствующий вершине из следующей группы, и в столбце, соответствующем этой вершине, отсутствуют ненулевые элементы в строках отмеченной группы, то отмеченную вершину можно объединить с проанализированной вершиной из следующей группы.

Формально это условие выглядит так:

Пусть v_i^1 – отмеченная вершина группы G_1, v_j^2 – вершина из следующей группы G_2, v_{ij} – элементы матрицы смежности, стоящие на пересечении строк вершин группы G_1 и j-го столбца, тогда:

$$v_i^1 = v_i^1 + v_j^2, \ если \left(v_j^1 \neq 0\right) \cap \left(v_{ij} = 0\right).$$

5) Повторить четвертый шаг для всех вершин отмеченной группы.

6) Повторить шаги 3-5 для всех следующих групп, кроме последней.

7) Повторять алгоритм до тех пор, пока не останется ни одного объединения.

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

Границы минимальной ширины информационного графа. Данный алгоритм позволяет существенно уменьшить высоту и ширину информационного графа. Остается открытым вопрос о верхней границе оценки минимальной ширины графа, той границе, которую на практике можно достигнуть всегда.

При оптимизации информационного графа по ширине передвигать вершины графа можно только в направлении от входных вершин к выходным, поэтому после окончания второй части алгоритма

можно рассчитать уточненную оценку минимальной ширины с учетом числа групп:

$$d' = \max_{1 \le k \le m} \left\lceil \frac{\sum_{i=i_k}^{n} d_i}{s-k+1} \right\rceil, \qquad (1)$$

где m – число групп, i_k – номер первой вершины в k-й группе.

Формула (1) является более приближенной к практике оценкой минимальной ширины графа, но и она не является верхней границей. Если выходная вершина одна, то как минимум одна группа (последняя) будет состоять из одной вершины. Следовательно, плотность на других ярусах будет выше. Аналогичный вывод можно сделать для входных вершин. Поэтому рассчитывать верхнюю оценку минимальной ширины графа следует с учетом числа выходных вершин.

Обозначим $\Delta_{\text{вх}} = d' - n_{\text{вх}}$ – разность между шириной графа и числом входных вершин, где $n_{\text{вх}}$ – количество входных вершин. Аналогично, $\Delta_{\text{вых}} = d' - n_{\text{вых}}$, где $n_{\text{вых}}$ – количество выходных вершин Рассмотрим возможные варианты:

1) $\Delta_{\text{вх}} > 0, \Delta_{\text{вых}} > 0$. Плотность остальных групп повышается на величину:

$$\Delta = \left\lceil \frac{\Delta_{\text{вх}} + \Delta_{\text{вых}}}{g-2} \right\rceil, \qquad (2)$$

где g – число групп, полученных в результате первой части алгоритма. Следовательно: $d' = d' + \Delta$.

2) $\Delta_{\text{вх}} > 0, \Delta_{\text{вых}} < 0$. Количество выходных вершин больше минимальной ширины группы. Плотность последней группы уменьшить нельзя, поэтому: $d' = n_{\text{вых}}$.

3) $\Delta_{\text{вх}} < 0, \Delta_{\text{вых}} > 0$. Количество входных вершин больше минимальной ширины группы. Специфика информационного графа такова, что, в зависимости от информационных связей между первой и последующими группами, в результате второй части алгоритма ширина первой группы может и уменьшиться, и остаться в первоначальном состоянии. Поэтому поведение оценки минимальной ширины в данном случае остается под вопросом.

4) $\Delta_{\text{вх}} < 0, \Delta_{\text{вых}} < 0$. Данный случай является композицией 2-го и 3-го вариантов. Поэтому: $d' = n_{\text{вых}}$.

С учетом того, что число выходных вершин уже учтено в формуле (2), от последних трех вариантов можно отказаться, оставив верхнюю оценку минимальной ширины в следующем виде:

$$d' = d' + \Delta, \qquad (3)$$

где:

$$\Delta = \begin{cases} \dfrac{\Delta_{\text{вх}} + \Delta_{\text{вых}}}{g-2}, (\Delta_{\text{вх}} \succ 0) \cap (\Delta_{\text{вых}} \succ 0) \\ 0, \textit{в остальных случаях} \end{cases} \qquad (4)$$

Если свести полученные результаты на одну числовую ось d, можно определить границы практической минимальной ширины информационного графа (рис. 1): Таким образом, после применения первой части метода оптимизации параллельного алгоритма по ширине можно определить границы минимальной ширины графа, и уже во время применения второй части метода оптимизации по ширине судить об эффективности его применения. В самом алгоритме, во второй части, в зависимости от архитектуры используемого вычислительного кластера, можно применять вместо теоретической оценки минимальной ширины d практическое значение d' или любое другое значение, диктуемое условиями решения задачи.

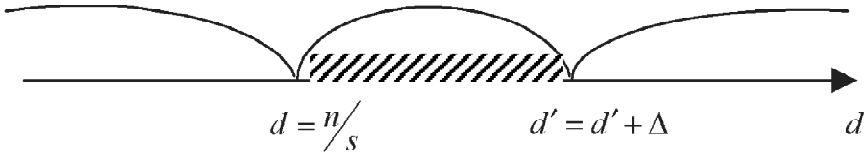


Рис. 1. Интервал практической минимальной ширины информационного графа.

Апробация алгоритма оптимизации. Для апробации алгоритмов оптимизации параллельных программ по ширине графа и времени выполнения алгоритмов, лежащих в основе программ, 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 (рис. 2);

✓ строить вручную произвольные ориентированные графы и задавать веса;

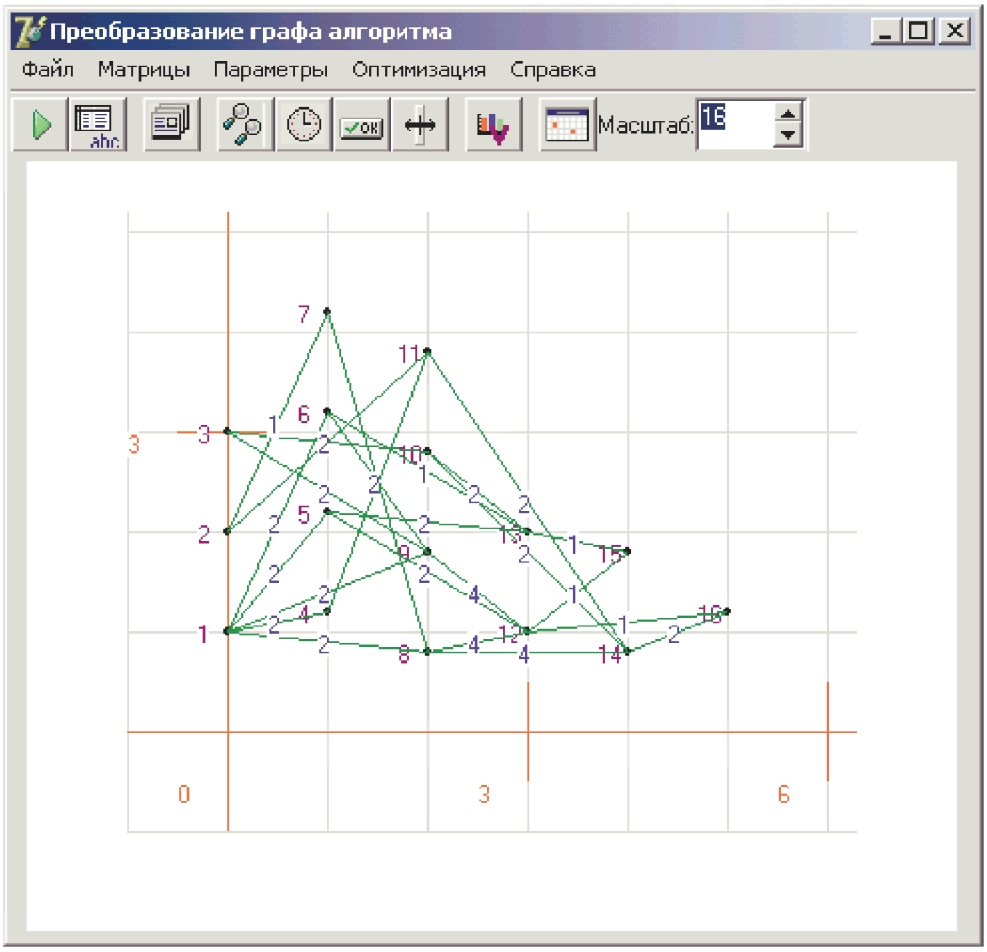


Рис. 2. Информационный граф алгоритма.

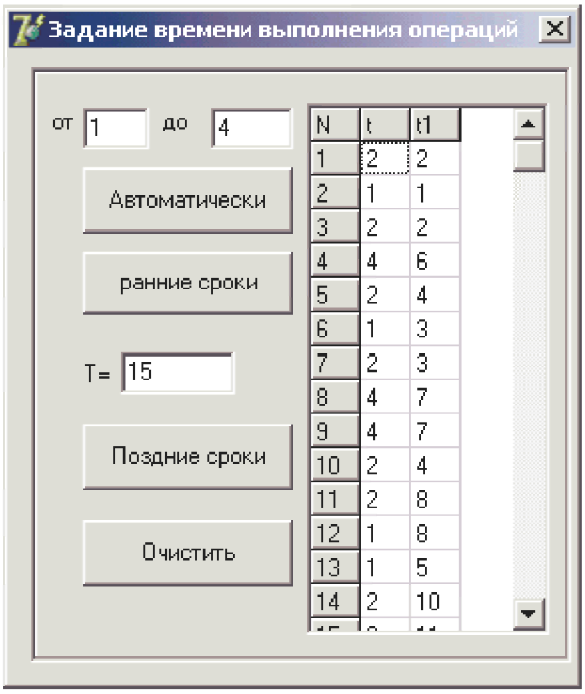
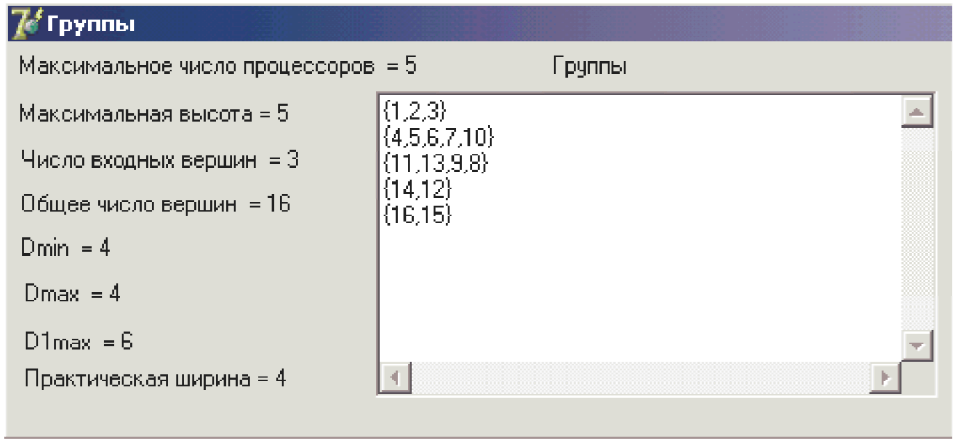


Рис. 3.Расчет ранних и поздних сроков выполнения операций



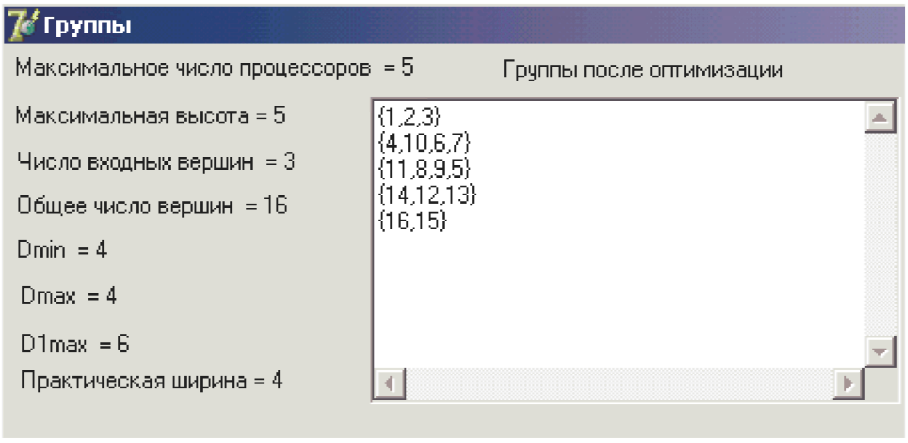


Рис. 4.Расчет дополнительных параметров и групп до и после оптимизации

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

✓ оптимизировать информационный граф по ширине (рис. 4);

✓ оптимизировать информационный граф по времени выполнения;

✓ оптимизировать информационный граф по ширине и времени выполнения комбинированным алгоритмом, полученным из объединения и модификации алгоритмов оптимизации;

✓ строить временные диаграммы (рис. 5).

На рис. 5а приведена временная диаграмма исходного информационного графа (рис. 2). Время выполнения алгоритма в соответствии с этим информационным графом равно 15 ед., число процессоров – 4. Последующие три диаграммы иллюстрируют работу алгоритмов оптимизации: по ширине (процессоров стало 4) (рис.5б), по времени (время сократилось до 13 ед.) (рис.5в);

✓ построение матриц смежности, следования, списков связности;

✓ вывод совокупностей операций, выполняемых каждым процессором.

Следует заметить, что не всегда удается избежать простоев процессоров и свести время работы к теоретическому минимальному времени.

По результатам исследования можно предложить следующую методику применения разработанных алгоритмов:

1) Разбить совокупность операций на группы с применением первой части алгоритма оптимизации информационного графа по ширине.

2) Вычислить теоретическую оценку минимальной ширины графа d.

3) Если d>n (где n – число процессоров), то по результатам применения шага 1 добавить дополнительные группы. Если d ≤ n, то скорректировать значение d с условиями применяемого вычислительного кластера.

4) Провести оптимизацию по ширине с помощью второй части алгоритма оптимизации информационного графа по ширине.

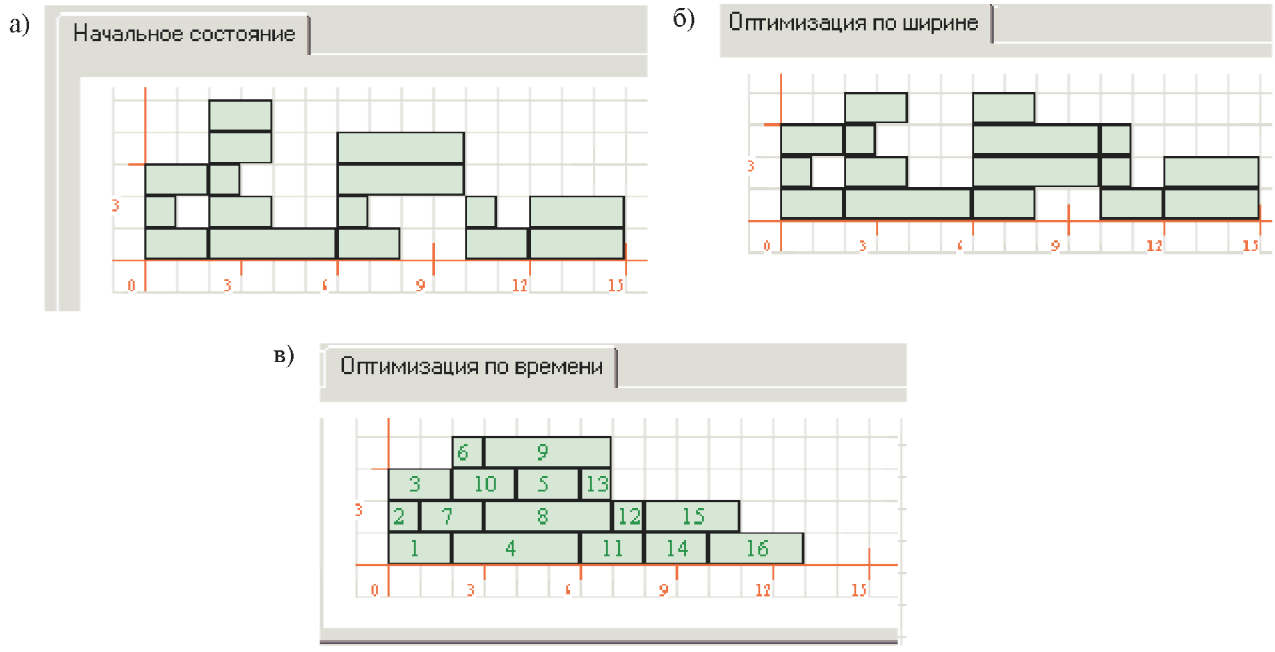


Рис. 5. Построение временных диаграмм.

5) Если в результате сохранилось по-прежнему много простоев, то применить к результатам шага 1 комбинированный метод оптимизации информационного графа по ширине и времени выполнения.

Заключение. Апробация полученных алгоритмов показала, что применение алгоритма оптимизации информационного графа по ширине, как правило, сокращая ширину графа, оставляет достаточно много простоев процессоров. Добиться наилучшего результата можно, только проведя после первого шага оптимизацию по времени или применив комбинированный метод. В зависимости от структуры информационного графа, в разных случаях превалирует тот или другой метод.

Литература

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

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