электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.051+519.168

Представление графовых моделей в системах параллельной обработки

***Головков А.А.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Компьютерные системы и сети»

Научный руководитель: Иванова Г.С., д.т.н., профессор
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Компьютерные системы и сети»
v.suzev@bmstu.ru

Теория графов находит широкое применение при решении различных комбинаторно-оптимизационных задач, существенную часть которых составляют NP-полные задачи. Алгоритмы их решения имеют высокую вычислительную сложность, как следствие требуют значительного времени, которое зависит от следующих взаимосвязанных факторов:

- архитектуры и характеристик вычислительных средств, для которых разрабатывается программа;
- способа машинного представления исходных и результирующих графов и эффективности реализации алгоритмов, встроенных в среду программирования.

Существенное увеличение быстродействия может обеспечить применение параллельных вычислительных систем. Такие системы, в отличие от обычных компьютеров, имеют некоторые особенности, которые связаны с архитектурой системы, в частности с организацией памяти и работой с ней.

Для представления графов в программных системах могут быть использованы матрицы смежности или инцидентности. Однако реализации на основе матриц при больших размерностях графов в реальных задачах приводят к неэкономному использованию памяти, так как в большинстве реальных задач эти матрицы сильно разрежены [1].

Соответственно более эффективным является аналитическое представление графов множествами вершин X и ребер U и множествами множеств их отображений по отношениям смежности FX, FU и инцидентности GX, GU.

В [1] рассмотрены различные варианты структур, используемые для реализации представлений графов:

- матрица;
- массив статических векторов;
- массив динамических векторов;
- список векторов;
- список списков;
- массив n-связных списков;
- список n-связных списков.

Указанные структуры широко применяются при разработке алгоритмов операций над графами. Исследования возможностей реализации параллельных алгоритмов и анализ применимости существующих структур данных, привели к необходимости разработки структуры данных для представления графа, ориентированной на параллельные вычислительные системы, с целью решения различных проблем, связанных с параллельным программированием, например:

- планирование работы – выделение для каждого потока отдельных областей структуры представления графа, например вершины или ребра;
- одновременное использование памяти – возникновение конфликтов по данным: чтение после записи, запись после чтения, запись после записи.

Структура должна быть единой, централизованной и минимально иерархичной: вершины, ребра, их идентификаторы и веса, инцидентность вершин и ребер должны быть представлены в виде общей структуры с явной связностью, прозрачным доступом и сведенным к минимуму дублированием данных.

Множества, определяющие отношения инцидентности и смежности, вершины и ребра в графе, могут быть представлены в виде контейнеров соответствующего типа. Для их реализации целесообразно использовать массивы. Доступ к элементам контейнера будет осуществляться по индексу из разных потоков. Массив позволит эффективно реализовать атомарные операции добавления и удаления элемента, но привнесет трудности, связанные с динамическим изменением размера массива. За счет начального резервирования памяти большого объема для массива эти трудности могут быть преодолены.

Рассмотрим структуру данных для представления гиперграфа с количеством вершин n, количеством ребер m. Каждая вершина и ребро имеют идентификатор id от 0 до

n - 1 и от 0 до m - 1 соответственно. Пример такого графа для n = 4 и m = 2 представлен на рисунке 1.

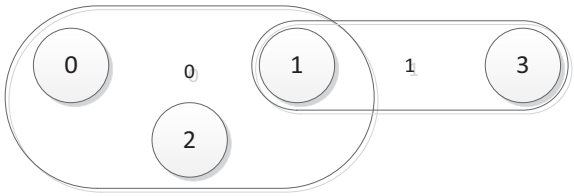

Рис. 1. Гиперграф

Гиперграф будет представлен структурой, показанной на рисунке 2. Она состоит из 2-ух указателей vertices и edges на массивы, содержащие указатели на вершины и ребра графа соответственно. Размеры этих массивов – verticesCount и edgesCount, равные n и m. Вершина или ребро графа описываются структурой, содержащей поля:

- id – идентификатор вершины или ребра;
- weight – вес вершины или ребра;
- connections – указатель на массив указателей инцидентных вершин или ребер;
- size – размер массива connections.

За счет возможности реализации отношений вершин и ребер как многие ко многим и их двусторонней связности такое представление графа может быть расширено на реализацию любых графовых структур.

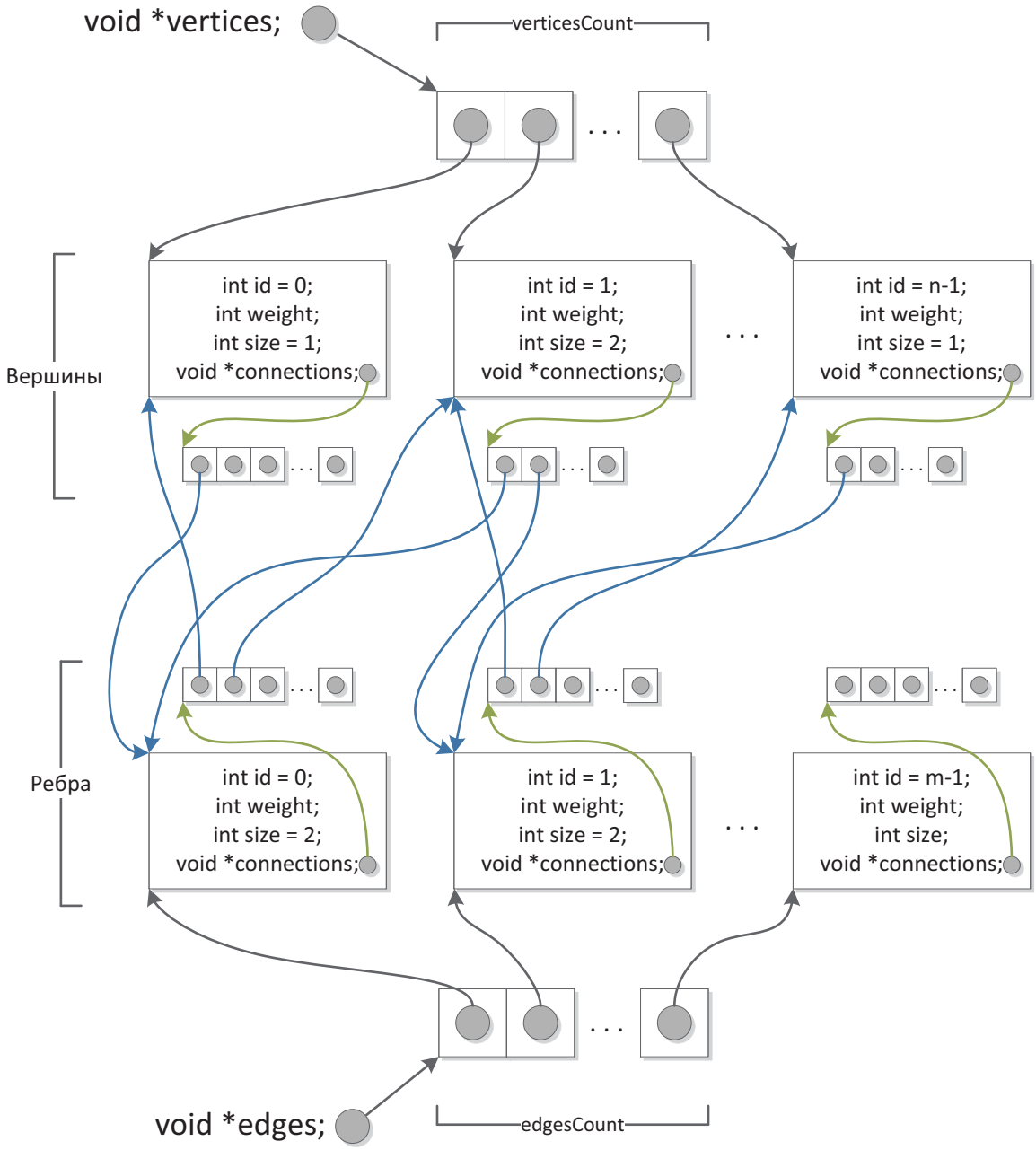

Рис. 2. Структура данных для представления графа

Данная структура имеет ряд преимуществ по сравнению с другими реализациями:

- ориентированность на параллельную реализацию алгоритмов работы со структурой — явно выделены структуры вершин и ребер, что дает возможность работать со структурой графа нескольким потокам, которые будут использовать свою отдельную вершину или ребро;
- хорошая масштабируемость — представление вершины и ребра в виде структур позволяет при необходимости добавлять различные поля в структуру при описании алгоритмов;

- реализация массивами указателей на вершины и ребра, по сравнению с реализацией массивами самих вершин и ребер позволяет сократить время, необходимое для сдвига элементов массивов при удалении элемента из середины.

Как недостаток можно выделить отсутствие явного представления отношений смежности FX и FU, но они могут быть получены из отношений инцидентности GX и GU.

Любой алгоритм решения задачи, которая требует применение теории графов, представляет собой совокупность вычислений характеристик графа и операций преобразования исходного графа в граф результата в соответствии с различными условиями. Операции над графами [2-4] обычно составляют большую часть тела разрабатываемого алгоритма.

Анализ операций показал наличие высокой степени параллелизма, в общем случае равной количеству вершин или ребер графа. Следовательно, для получения максимального ускорения обработки графовой модели необходимо использовать параллельную вычислительную систему с большим числом выполняемых параллельно задач. Таким требованиям подходят вычислительные системы с архитектурой CUDA [5].

Все потоки графического процессора CUDA могут работать только с памятью графического процессора, которая на сегодняшний день достигает нескольких гигабайт. Для того чтобы исключить копирование графа, определенного описанной выше структурой, из оперативной памяти в память GPU и обратно, она должна быть полностью расположена в памяти видеокарты.

Рассчитаем объем видеопамяти необходимый для хранения гиперграфа. Учитывая, что размер указателя равен 4 байтам, объем памяти для размещения всего графа в структуре, описанной выше, в байтах составляет:

$$M = 16 + 4(n + m) + 4(4 + size_n)\mathrm{n} + 4(4 + size_m)\mathrm{m}, \quad (1)$$

где 16 – суммарный размер указателей на массивы vertices и edges и переменных verticesCount и edgesCount;

n – количество вершин графа;

m – количество ребер графа;

$4(n + m)$– размер массивов vertices и edges;

$size_n$ – средняя степень вершин графа;

$size_m$ – средняя степень ребер графа;

$4(4 + size_n)n$ – размер структур вершин;

$4(4 + size_m)m$ – размер структур ребер.

Полагая степени вершин $size_n$ и ребер $size_m$ равными 10 и 50 соответственно, (1) примет вид:

$$M = 16 + 60n + 220m. \quad (2)$$

График функции (2) представлен на рисунке 3. При объеме памяти графического процессора 4 ГБ предложенная структура позволит хранить графы с количеством вершин и ребер более 15 млн.

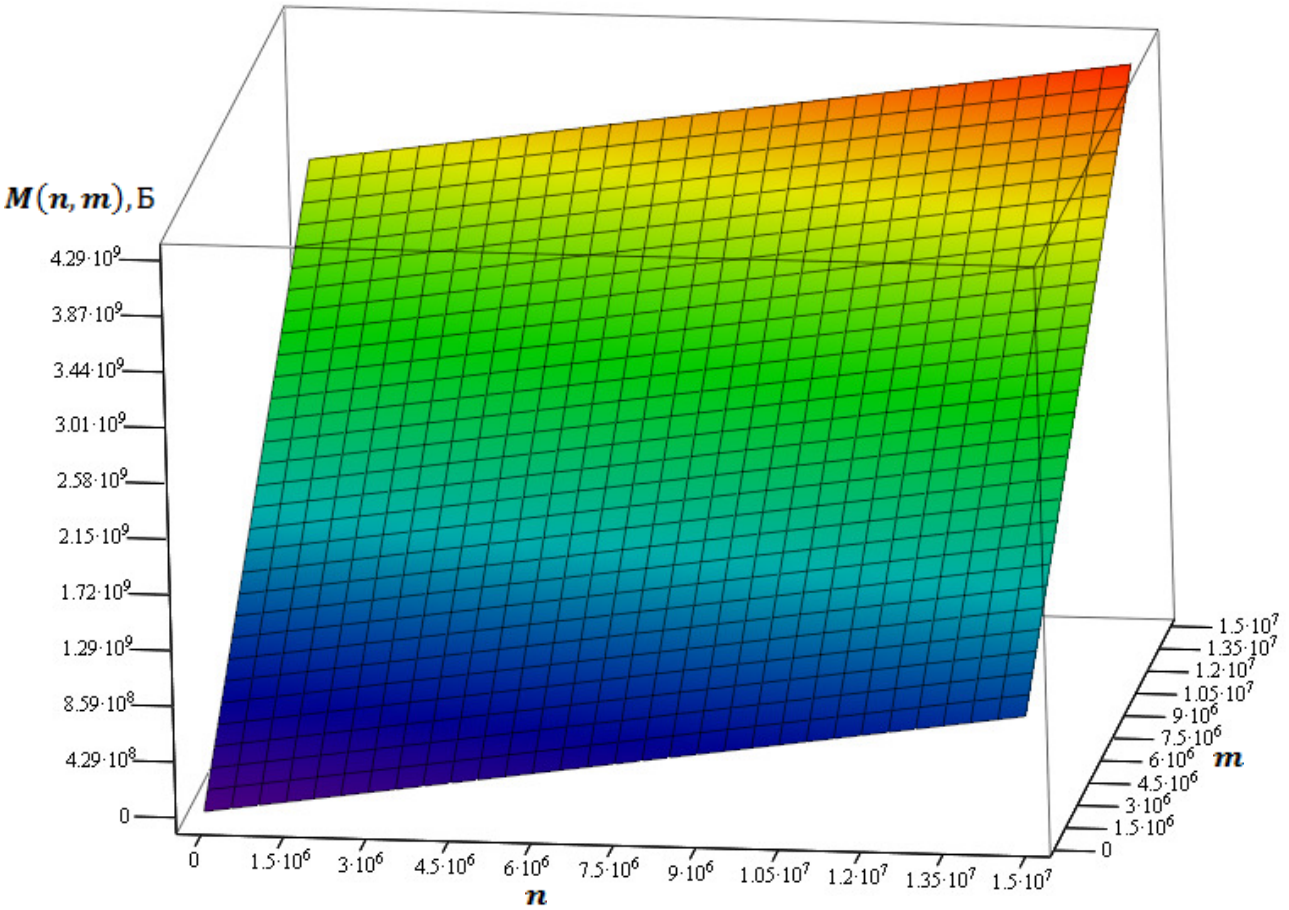

Рис. 3. График функции $M(n, m)$

Предложенная модель данных для представления графа полностью отвечает сформулированным требованиям к структуре, ориентированной на параллельные вычислительные системы, и может успешно применяться при реализации параллельных алгоритмов операций над графами в параллельных вычислительных системах. Это позволит существенно сократить время выполнения программ, обрабатывающих графы с большим количеством вершин и ребер и требующих значительных вычислительных мощностей.

Список литературы

1. Овчинников В.А., Иванова Г.С., Ничушкина Т.Н. Выбор структур данных для представления графов при решении комбинаторно-оптимизационных задач // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2001. № 2(43). С. 39–51.
2. Овчинников В.А. Операции над ультра и гиперграфами для реализации процедур анализа и синтеза структур сложных систем // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2009. № 10. Режим доступа: http://technomag.bmstu.ru/doc/132769.html (дата обращения 18.04.2014).
3. Овчинников В.А. Операции над ультра и гиперграфами для реализации процедур анализа и синтеза структур сложных систем (часть 2) // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2009. № 11. Режим доступа: http://technomag.bmstu.ru/doc/133223.html (дата обращения 18.04.2014).
4. Овчинников В.А. Операции над ультра и гиперграфами для реализации процедур анализа и синтеза структур сложных систем (часть 3) // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2009. № 12. Режим доступа: http://technomag.bmstu.ru/doc/134335.html (дата обращения 18.04.2014).
5. Rob Farber. CUDA Application Design and Development. Waltham: Morgan Kaufmann, MA, USA. 2011. 336 p.