Попов Даниил Владимирович, Юго-Западный государственный университет,
аспирант кафедры вычислительной техники

Анализ эффективности применения технологий GPGPU при реализации эвристических алгоритмов в задачах на графах

Задачи дискретной комбинаторной оптимизации делятся на два больших класса: класс сложности P и класс сложности NP. К классу сложности P относят задачи, для решения которых известны алгоритмы, обеспечивающие получение оптимального решения за полиномиальное время, а к классу NP – те задачи, для которых такие алгоритмы не известны или доказана невозможность их построения. На практике для решения задач класса NP и задач большой размерности класса P применяются эвристические методы. Они не гарантируют нахождения решения (и его оптимальности), но имеют гораздо меньшую временную сложность. Среди эвристических методов выделяют итерационные, выполняющие построение ограниченного конечного множества решений с заданным ограничением и выбирающие лучшее из них. Для некоторых методов становится возможным применение технологий параллельных вычислений, для получения множества решений, среди которых будет выбрано лучшее.

CUDA и OpenCL представляют собой два разных интерфейса для программирования графических процессоров. OpenCL – это открытый стандарт, который можно использовать для программирования процессоров, графических процессоров и других устройств от разных производителей, а CUDA – только для графических процессоров NVIDIA. Хотя OpenCL является кроссплатформенной технологией, поддержка различных типов устройств может негативно влиять на производительность.

В данной работе основное внимание уделено сравнению производительности CUDA и OpenCL при реализации некоторых итерационных эвристических алгоритмов в задачах на графах.

В качестве тестовой задачи была взята широко известная задача поиска кратчайшего пути $L = [a_{i1}, a_{i2}, \dots, a_{in}]$ в графе $G = \langle A, V \rangle$, где $A = \{a_1, a_2, \dots, a_N\}$ – множество вершин графа, $N = \mid A \mid$ – количество вершин в графе, $V = \{v_1, v_2, \dots, v_M\} \subseteq V \times V$ – множество дуг, а $M = \mid V \mid$ – количество дуг. Тогда плотность графа $d = \frac{M}{N(N-1)}$. Каждой дуге приписана длина $l(v_i) > 0$. Целевой функцией будет являться длина пути $l(L) = \sum_{v_i \in V_L} l(v_i) \to min$, где $V_L = \{(a_{i1}, a_{i2}), (a_{i2}, a_{i3}), \dots, (a_{iL-1}, a_{iL})\} \subseteq V$ – множество образующих путь дуг [1].

Для тестирования были выбраны метод случайного (англ. Random Search, сокр. RS) [2] и взвешенного случайного перебора (англ. Weighted Random Search, сокр. WRS) [3], метод муравьиной колонии (англ. Ant Colony, сокр. AC)[2,4]. Хотя во время поиска решения методом муравьиной колонии после каждой итерации необходимо выполнять обновление матрицы феромона, есть возможность параллельно строить решения, которые находят «муравьи». Для сравнения были разработаны параллельные версии данных алгоритмов (англ. Parallel, сокр. P в имени метода).

Целью организованного вычислительного эксперимента был анализ среднего времени нахождения решения $t = \frac{1}{K}\sum_{i=1}^{K} t_i$, где $K=1000$ – объем выборки тестовых примеров, t_i – время нахождения решения для i-ого элемента выборки, при равном количестве запущенных потоков на GPU.

Эксперимент проводился с использованием CPU Intel Core i3-4330 и GPU Nvidia GTX 750 Ti. Поиск решений проводился для графов с количеством вершин 50 и плотностью графа в диапазоне от 0.01 до 1.

Ниже приведены результаты вычислительного эксперимента для количества запускаемых потоков на GPU $C=256$. При этом, ядра CUDA имели конфигурацию 16 блоков по 16 потоков в каждом.

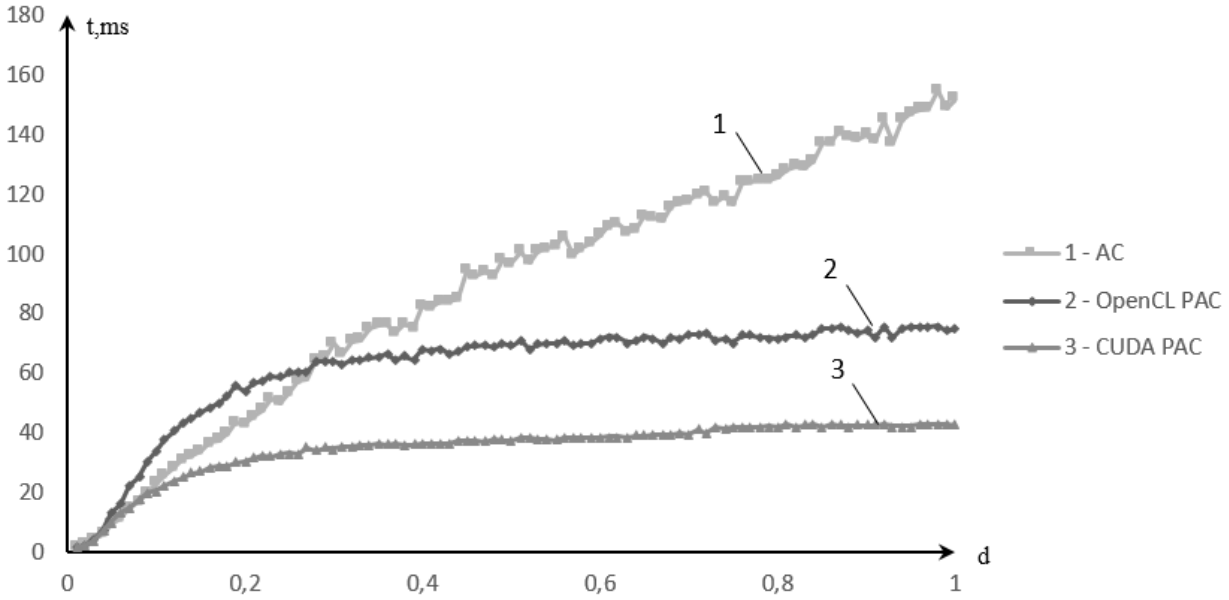

Рисунок 1 – Зависимости времени получения решения от плотности графа для алгоритма муравьиной колонии

На рисунке 1 представлены результаты эксперимента для метода муравьиной колонии. Как можно увидеть, использование технологии CUDA позволяет добиться уменьшения временных затрат для данного до 4,3 раз, а использование OpenCL – до 2,7 раза. При этом, средний выигрыш во времени нахождения решения за счет использования технологии CUDA по сравнению с OpenCL составляет 1,7 раза.

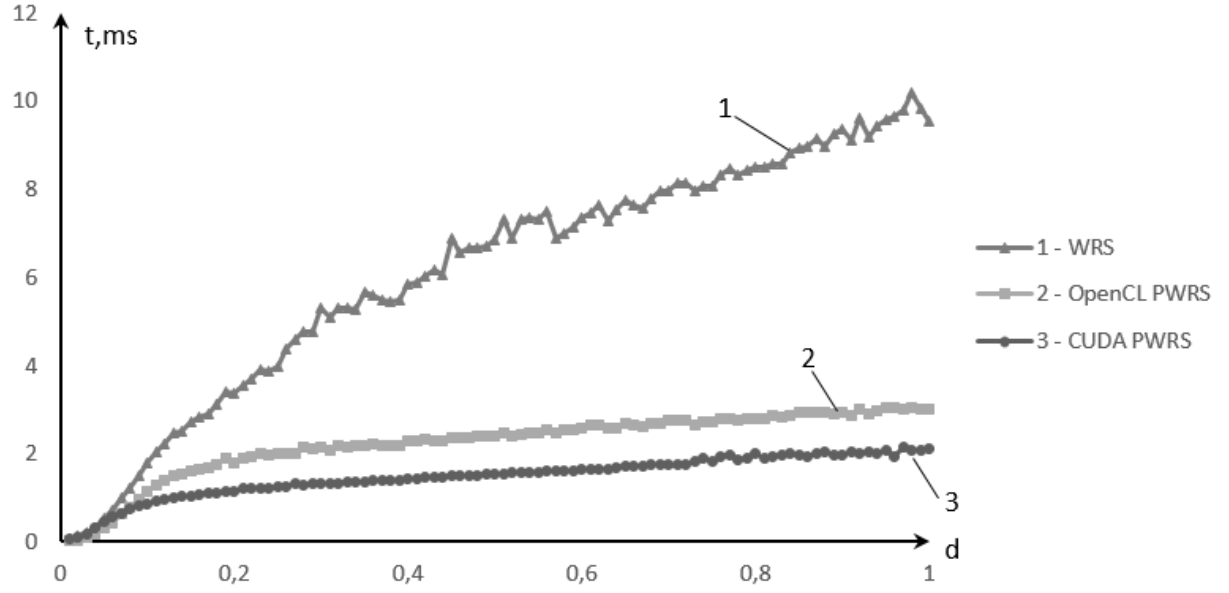

Рисунок 2 – Зависимости времени получения решения от плотности графа для взвешенного случайного перебора

Результаты эксперимента для случайного и взвешенного случайного переборов представлены на рисунке 2 и 3. При реализации метода взвешенного случайного перебора CUDA дает выигрыш в среднем в 1,47 раза по сравнению с OpenCL. А для метода случайного перебора выигрыш может достигать 1,85 раза при плотности графа d=0,75, и составляет 1,6 раза в среднем.

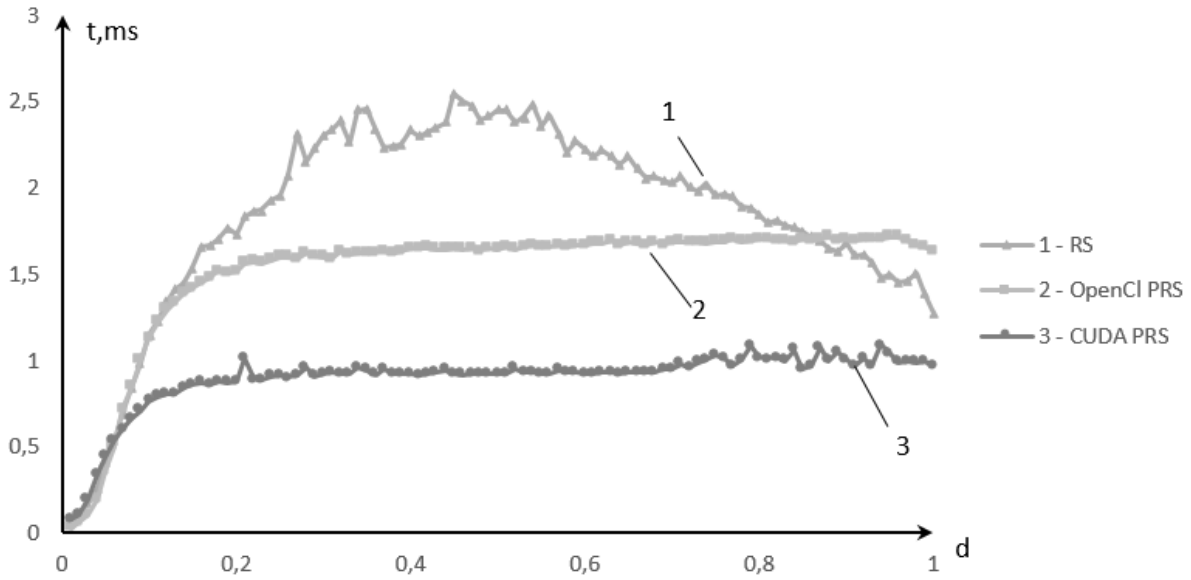

Рисунок 3 - Зависимости времени получения решения от плотности графа для случайного перебора

При увеличении количества потоков до 1024 среднее отставание OpenCL от CUDA остается на уровне 1,4 раза в среднем для методов случайного и случайного взвешенного перебора, а для метода муравьиной колонии достигает в среднем 3,5 раз. Так, для алгоритма муравьиной колонии при плотности графа d = 0,7 использование GPU с поддержкой технологии OpenCL обеспечивает выигрыш по сравнению с реализацией на CPU примерно в 3,1 раза (против 16,2 для технологии CUDA).

Полученные данные позволяют сделать вывод о том, что использование технологии CUDA дает больший выигрыш в скорости выполнения, по сравнению с технологией OpenCL. Однако, технология CUDA может применяться только на видеокартах производства фирмы Nvidia, в то время как OpenCL является кроссплатформенной технологией.

Список литературы

1. Ватутин Э.И., Титов В.С., Емельянов С.Г. Основы дискретной комбинаторной оптимизации. М.: АРГАМАК-МЕДИА, 2016. 270 с.

2. Ватутин Э.И., Титов В.С. Анализ результатов применения алгоритма муравьиной колонии в задаче поиска пути в графе при наличии ограничений // Известия Южного федерального университета. Технические науки. 2014. № 12 (161). С. 111–120.

3. Ватутин Э.И., Колясников Д.В., Мартынов И.А., Титов В.С. Метод случайного перебора в задаче построения разбиений граф-схем параллельных алгоритмов // Многоядерные процессоры, параллельное программирование, ПЛИС, системы обработки сигналов. Барнаул: Барнаул, 2014. С. 115–125.

4. Ватутин Э.И., Дремов Е.Н., Мартынов И.А., Титов В.С. Метод взвешенного случайного перебора для решения задач дискретной комбинаторной оптимизации // Известия ВолГТУ. Серия: Электроника, измерительная техника, радиотехника и связь. № 10 (137). Вып. 9. 2014. С. 59–64.

5. Dorigo M. Optimization, Learning and Natural Algorithms // PhD thesis. Politecnico di Milano, Italie, 1992.