# Исследование эффективности архитектуры CUDA для аппроксимации множества Парето с помощью метода роя частиц

А.Э. Антух, А.П. Карпенко, А.С. Семенихин

МГТУ им. Н.Э. Баумана

Во многих практически значимых случаях при решении задачи многокритериальной оптимизации предварительно целесообразно построить аппроксимацию множества Парето этой задачи. Рассматривается комбинация известного метода приближенного построения множества Парето «недоминируемая сортировка» и метода глобальной оптимизации роем частиц. Целью работы является исследование эффективности указанной комбинации методов при их реализации на графических процессорных устройствах с архитектурой CUDA.

## 1. Введение

Использование графических ускорителей (ГПУ) для научных расчетов началось относительно недавно [1]. Во многих прикладных областях до сих пор остается открытым вопрос об эффективности ГПУ по сравнению с классическими компьютерными системами. Целью данной работы является исследование эффективности ГПУ с архитектурой CUDA для задачи аппроксимации множества Парето с помощью метода роя частиц (PSO).

Основная масса публикаций, посвященных CUDA-вычислениям, ориентирована на задачи, обладающие явным параллелизмом по данным [2]. Такие задачи хорошо распараллеливаются и позволяют получать существенный прирост производительности вычислений по сравнению с последовательными вычислениями на хост-ЭВМ. В задачах многокритериальной оптимизации не всегда присутствует параллелизм по данным и эффективность решения таких задач на ГПУ с архитектурой CUDA мало освещена в литературе. Исследование эффективности CUDA-вычислений при решении задачи однокритериальной оптимизации выполнено, например, в работе [3].

В разделе 2 работы приведена математическая формулировка задачи многокритериальной оптимизации и дано определение множества Парето. Раздел 3 содержит краткое описание используемого алгоритма метода роя частиц. В разделе 4 приведено описание особенностей реализации алгоритма, а также дана постановка тестовой задачи многокритериальной оптимизации. Раздел 5 содержит результаты экспериментов и оценку эффективности алгоритма. В заключении подведены итоги работы и определены направления ее развития.

## 2. Постановка задачи

Задана совокупность частных критериев оптимальности $\phi_1(X), \phi_2(X), ..., \phi_m(X)$, которые образуют векторный критерий $\Phi(X)$. Здесь $X$ - $n$-мерный вектор варьируемых параметров [4]. Ставится задача минимизации каждого из указанных критериев в одной и той же области допустимых значений вектора варьируемых параметров $D_X \in \Pi \cap D$, где $\Pi = \left\{X \middle| x_i^- \le x_i \le x_i^+, i \in [1,n]\right\}$ - «технологический» параллелепипед, $D = \left\{X \middle| g_1(X) \ge 0, g_2(X) \ge 0, ...\right\}$. Здесь $g_1(X), g_2(X), ...$ - ограничивающие функции. Условно задача многокритериальной оптимизации записывается в виде

$$\min_{X \in D_X} \Phi(X) = \Phi(X^*). \quad (1)$$

Векторный критерий оптимальности $\Phi(X)$ выполняет отображение множества $D_X$ в некоторое множество $D_\Phi$ пространства критериев, которое называется множеством достижимости

задачи (1). Введем на множестве $D_\Phi$ отношение предпочтения. Будем говорить, что вектор $\Phi^1 \in D_\Phi$ предпочтительнее вектора $\Phi^2 \in D_\Phi$ (или вектор $\Phi^1$ доминирует вектор $\Phi^2$), и писать $\Phi^1 \succ \Phi^2$, если среди равенств и неравенств $\phi_k(X^1) \geq \phi_k(X^2), k \in [1:m]$ имеется, хотя бы одно строгое неравенство. Выделим из множества $D_\Phi$ подмножество точек $D_\Phi^*$ (фронт Парето), для которых нет более предпочтительных точек. Множество $D_X^* \in D_X$, соответствующее множеству $D_\Phi^*$, называется множеством Парето [4]. Таким образом, если $X \in D_X^*$, то $\Phi(X) \in D_\Phi^*$.

Если $\Phi(X_1) \succ \Phi(X_2)$, то будем говорить, что вектор $X_1$ предпочтительнее вектора $X_2$ (или вектор $X_1$ доминирует вектор $X_2$), и писать $X_1 \triangleright X_2$.

Ставится задача приближенного построения множества Парето $D_X^*$ или, что то же самое, фронта Парето $D_\Phi^*$, задачи многокритериальной оптимизации (1).

## 3. Методы роя частиц и недоминируемой сортировки

Множество частиц обозначим $\mathbf{P} = \{P_i, i \in [1:N]\}$, где $N$ – число частиц в рое (размер популяции). В дискретный момент времени $t \in [0:T]$ координаты частицы $P_i$ определяются вектором $X_{i,t} = (x_{i,t,1}, x_{i,t,2}, ..., x_{i,t,n})$, а ее скорость – вектором $V_{i,t} = (v_{i,t,1}, v_{i,t,2}, ..., v_{i,t,n})$; $T$ - число итераций. Начальные координаты и скорости частицы $P_i$ равны $X_{i,0} = X_i^0$, $V_{i,0} = V_i^0$ соответственно.

Итерации в каноническом методе PSO выполняются по схеме [5]

$$V_{i,t+1} = \alpha V_{i,t} + U_1[0,\beta] \otimes \left(X_{i,t}^b - X_{i,t}\right) + U_2[0,\gamma] \otimes \left(X_{g,t} - X_{i,t}\right), \quad (2)$$

$$X_{i,t+1} = X_{i,t} + V_{i,t+1}. \quad (3)$$

Здесь $U[a,b]$ представляет собой $n$-мерный вектор псевдослучайных чисел, равномерно распределенных в интервале $[a,b]$; $\otimes$ - символ покомпонентного умножения векторов; $X_{i,t}^b$ - вектор координат частицы $P_i$ с наилучшим значением целевой функции $\Phi(X)$ за все время поиска $[0:t]$; $X_{g,t}$ - вектор координат соседней с данной частицы с наилучшим за время поиска $[0:t]$ значением целевой функции $\Phi(X)$; $\alpha, \beta, \gamma$ - свободные параметры алгоритма. Важнейшее в методе PSO понятие соседства частиц зависит от используемой топологии соседства и определено, например, в работе [5]. В процессе итераций вектор $X_{i,t}^b$ образует так называемый собственный путь (private guide) частицы $P_i$, а вектор $X_{g,t}$ - локальный путь (local guide) этой частицы.

Канонический метод роя частиц ориентирован на решение задач однокритериальной оптимизации. Для задачи многокритериальной оптимизации (1) используем модификацию этого метода, которая называется метод MOPSO (Multi Objective Particle Swarm Optimization) [6].

Важной частью метода MOPSO является определение глобально-лучшей (в смысле формулы (1)) частицы для каждого индивидуума в популяции. В силу специфики задачи многокритериальной оптимизации глобально-лучшая частица отыскивается на множестве Парето.

Псевдокод метода MOPSO представлен на рисунке 1. На каждой итерации в процессе выполнения шаге 3 метода происходит обновление архива частиц $A_t$. Функция *Update* выполняет сравнение частиц текущего поколения с недоминируемыми частицами из архива $A_t$ и определяет те частицы, которые необходимо добавить в архив, а также те частицы, которые следует в архиве заменить. Функция реализует метод недоминируемой сортировки, предложенный в работе [7].

Выбор глобально лучшей частицы осуществляет функция *FindGlobalBest.* Существует несколько способов реализации этой функции. В данной работе используется метод «меняющихся» соседей» Хью и Эберхарта [6]. Рассмотрим суть этого метода на примере задачи двухкритериальной оптимизации. Поиск глобально лучшей частицы для каждой частицы популяции

осуществляется в пространстве критериев следующим образом. Сначала вычисляем расстояние от частицы $P_i$ до других частиц, содержащихся в архиве $A_t$, используя значения первого («фиксированного») критерия оптимальности $\phi_1(X)$. Таким образом, для частицы $P_i$ находим $k$ ее ближайших локальных соседей. Затем, используя второй критерий $\phi_2(X)$, из числа указанных $k$ соседей находим наилучшую частицу для частицы $P_i$, которая и полагается глобально лучшей частицей для частицы $P_i$.

**Шаг 1:** t=0
**Шаг 2:** Инициализация популяции $\mathbf{P}_t$ и архива частиц $A_t$
**For** i=1 **to** N
$X_{i,0} \coloneqq X_i^0$ ; $V_{i,0} \coloneqq V_i^0$ ; $X_{i,0}^b \coloneqq X_i^0$
End;
$A_t \coloneqq \{\}$ ;
**Шаг 3:** $A_{t+1} \coloneqq Update(\mathbf{P}_t, A_t)$ ;
**Шаг 4:**
**For** i=1 **to** N
$X_{g,t} \coloneqq FindGlobalBest(A_{t+1}, X_{i,t})$ ;
**For** j=1 **to** n
$v_{i,t+1,j} \coloneqq \alpha v_{i,t,j} + U_1[0,\beta] \otimes \left(x_{i,t,j}^b - x_{i,t,j}\right) + U_2[0,\gamma] \otimes \left(x_{g,t,j} - x_{i,t,j}\right)$;
$x_{i,t+1,j} \coloneqq x_{i,t,j} + v_{i,t+1,j}$
**End;**
**If** ( $X_{i,t+1} \rhd X_{i,t}^b$ ) $X_{i,t+1}^b \coloneqq X_{i,t+1}$ **Else** $X_{i,t+1}^b \coloneqq X_{i,t}^b$
**End;**
**Шаг 5:** **If** (Критерий останова=false)
t:=t+1;
**GOTO** Шаг 3
End.

**Рис. 1**. Псевдокод метода MOPSO

Итерации продолжаются до тех пор, пока множество недоминируемых решений не перестанет меняться, либо до достижения заданного числа итераций.

## 4. Реализация алгоритма и тестовая задача

Была рассмотрена следующая двухмерная двухкритериальная тестовая задача, точные фронт и множество Парето для которой известны:

$$\begin{cases} \phi_1(X) = x_1^2 + x_2^2, \\ \phi_2(X) = (x_1 - 1)^2 + (x_2 - 1)^2; \end{cases} \tag{4}$$

$$D_X = \{X \mid -5 \le x_i \le 5, i = 1,2\}. \tag{5}$$

Исследование плотности покрытия множества Парето и фронта Парето задачи (4, 5) при использовании метода MOPSO рассмотрено в работе [8]. В данной работе исследуется эффективность указанного метода при его реализации на ГПУ.

Исследование выполнено с использованием в качестве хост-ЭВМ персональной ЭВМ, управляемой операционной системой Windows XP, в следующей конфигурации: процессор - Intel Pentium Dual E2160, 1,8 ГГц; оперативная память - 2 ГБ. При вычислениях только с использованием хост-процессора было использовано одно ядро. В качестве ГПУ использована плата NVidia GeForce 8500GT, содержащая 16 потоковых процессоров. Архитектура CUDA рассмотрена, например, в работе [9].

В практически значимых задачах многокритериальной оптимизации основной объем вычислений связан с вычислениями значений частных критериев оптимальности. Например, если

речь идет о задаче оптимизации сложной динамической системы, каждое такое вычисление требует численного интегрирования большой системы обыкновенных дифференциальных уравнений. Поэтому в работе принята следующая схема организации вычислений: метод MOPSO реализует хост-процессор системы, а вычисления значений частных критериев оптимальности – ГПУ.

Эффективность параллельных вычислений оценивается с помощью ускорения

$$S = \frac{T_{CPU}}{T_{GPU}}, \tag{6}$$

где $T_{ЦПУ}$ – время решения задачи с использованием только хост-процессора системы, $T_{GPU}$ – аналогичное время при решении задачи с использованием ГПУ. Подчеркнем, что под ускорением $S$ в данном случае понимается повышение производительности вычислений при использовании ГПУ относительно производительности вычислений, производимых только на хост-процессоре.

Важная составная часть исследования – исследование зависимости ускорения (6) от суммарной вычислительной сложности частных критериев оптимальности. Для моделирования разных вычислительных сложностей критериев (4) использовано их многократное вычисление. В качестве меры вычислительной сложности критериев использован коэффициент сложности $C$, равный числу их вычислений.

## 5. Результаты экспериментов

Исследование выполнено при варьировании следующих параметров:

- число итераций $T = 30;\ 50$;
- коэффициент сложности $C = 1;\ 10;\ 100;\ 1000$;
- число частиц в популяции $N = 32;\ 64;\ 128;\ 256$.

Поскольку эффективность метода существенно зависит от начальных параметров частиц $X_i^0$, $V_i^0$ результаты исследования усреднены по этим начальным параметрам; $i \in [1:N]$.

Некоторые результаты исследования представлены в таблице 1. Таблица показывает, что, как и следовало ожидать, при невысокой вычислительной сложности частных критериев оптимальности распараллеливание вычислений не эффективно, поскольку в этом случае велика доля коммуникационных расходов. При увеличении вычислительной сложности критериев, а также числа частиц в популяции, ускорение вычислений достигает величины 23.

Таблицу 1 иллюстрируют рисунки 2, 3.

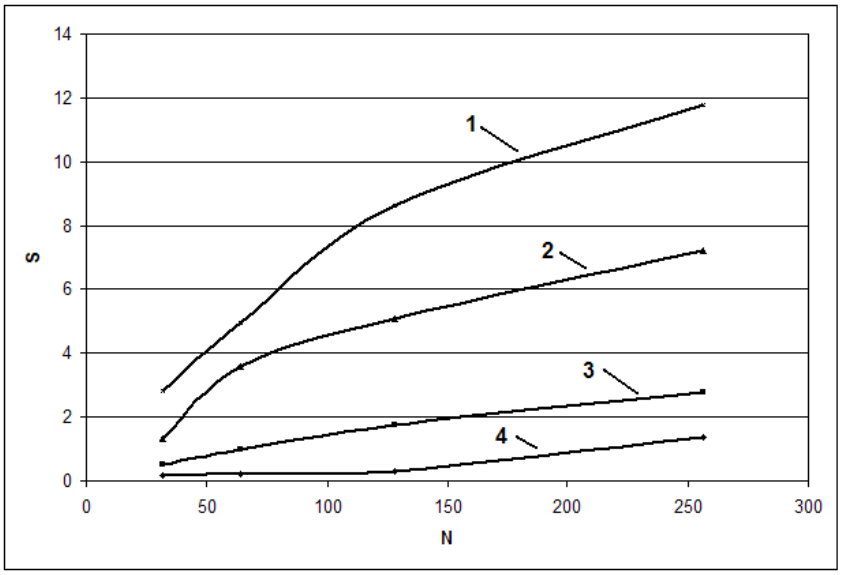


**Рис. 2**. Ускорение вычислений в функции числа частиц $N$ при числе итераций $T = 30$:
1 - $C = 1000$; 2 - $C = 100$; 3 - $C = 10$; 4 - $C = 1$

**Таблица 1.** Некоторые результаты исследования

| № | N | T | C | $T_{ЦПУ}$, с | $T_{GPU}$, с | S |
|---|---|---|---|---|---|---|
| 1 | 32 | 30 | 1 | 0,026 | 0,156 | 0,167 |
| 2 | 32 | 30 | 10 | 0,087 | 0,175 | 0,497 |
| 3 | 32 | 30 | 100 | 0,531 | 0,406 | 1,308 |
| 4 | 32 | 30 | 1000 | 6,922 | 2,468 | 2,805 |
| 5 | 32 | 50 | 1 | 0,062 | 0,21 | 0,295 |
| 6 | 32 | 50 | 10 | 0,297 | 0,265 | 1,121 |
| 7 | 32 | 50 | 100 | 2,39 | 0,656 | 3,643 |
| 8 | 32 | 50 | 1000 | 25,64 | 4,343 | 5,904 |
| 9 | 64 | 30 | 1 | 0,062 | 0,265 | 0,234 |
| 10 | 64 | 30 | 10 | 0,296 | 0,296 | 1,000 |
| 11 | 64 | 30 | 100 | 2,515 | 0,703 | 3,578 |
| 12 | 64 | 30 | 1000 | 23,328 | 4,703 | 4,960 |
| 13 | 64 | 50 | 1 | 0,147 | 0,359 | 0,409 |
| 14 | 64 | 50 | 10 | 0,643 | 0,421 | 1,527 |
| 15 | 64 | 50 | 100 | 6,556 | 1,14 | 5,751 |
| 16 | 64 | 50 | 1000 | 69,828 | 7,641 | 9,139 |
| 17 | 128 | 30 | 1 | 0,13 | 0,421 | 0,309 |
| 18 | 128 | 30 | 10 | 0,856 | 0,484 | 1,769 |
| 19 | 128 | 30 | 100 | 7,312 | 1,437 | 5,088 |
| 20 | 128 | 30 | 1000 | 79,641 | 9,25 | 8,610 |
| 21 | 128 | 50 | 1 | 0,54 | 0,656 | 0,823 |
| 22 | 128 | 50 | 10 | 2,498 | 0,796 | 3,138 |
| 23 | 128 | 50 | 100 | 28,504 | 2,609 | 10,925 |
| 24 | 128 | 50 | 1000 | 219,031 | 15,406 | 14,217 |
| 25 | 256 | 30 | 1 | 1 | 0,736 | 1,359 |
| 26 | 256 | 30 | 10 | 2,7 | 0,968 | 2,789 |
| 27 | 256 | 30 | 100 | 18,25 | 2,531 | 7,211 |
| 28 | 256 | 30 | 1000 | 268,36 | 22,812 | 11,764 |
| 29 | 256 | 50 | 1 | 3,36 | 1,234 | 2,723 |
| 30 | 256 | 50 | 10 | 12,44 | 1,5 | 8,293 |
| 31 | 256 | 50 | 100 | 81,437 | 4,45 | 18,300 |
| 32 | 256 | 50 | 1000 | 690,44 | 29,375 | 23,504 |

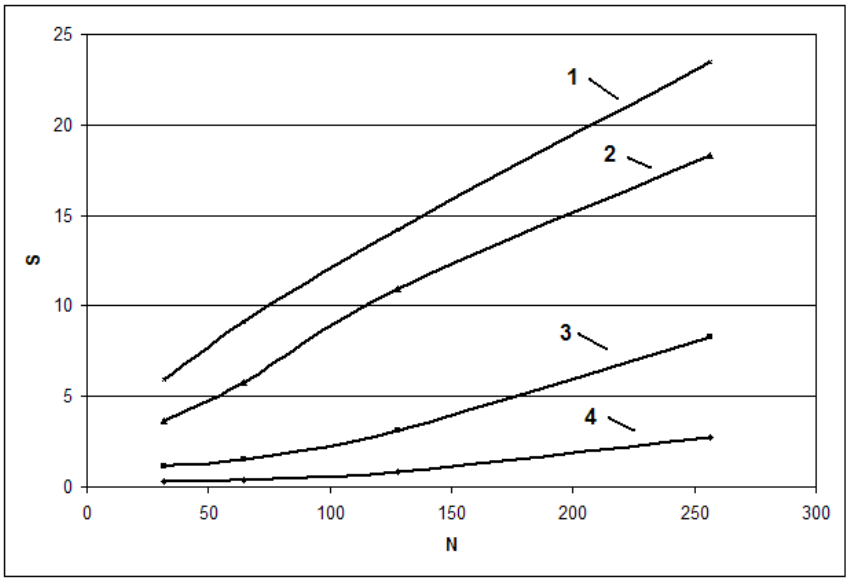


**Рис. 3**. Ускорение вычислений в функции числа частиц $N$ при числе итераций $T = 50$:
1 - $C = 1000$; 2 - $C = 100$; 3 - $C = 10$; 4 - $C = 1$

Отметим, что рисунки 2, 3 показывают практически линейный рост ускорения с ростом числа частиц *N*. Это обстоятельство позволяет надеяться на получение ускорения, превышающего максимально достигнутое в рассматриваемых экспериментах (равное 23) при дальнейшем увеличении числа частиц в популяции.

## 6. Заключение

Выполнено исследование эффективности комбинации метода MOPSO и метода недоминируемой сортировки при приближенном построении множества Парето в задаче многокритериальной оптимизации с помощью ГПУ с архитектурой CUDA. Исследование показало высокий потенциал использования графических ускорителей и библиотек для работы с ними (CUDA) для эффективного решения указанной задачи.

В развитие работы предполагается расширение класса тестовых задач многокритериальной оптимизации; определение для ГПУ различных архитектур оптимальных значений свободных параметров рассматриваемых методов, обеспечивающих максимальное ускорение вычислений; разработка методов теоретической оценки ускорения при заданных значениях свободных параметров методов и архитектуре ГПУ.

## Литература


1. Вычисления на GPU. http://www.nvidia.ru/page/gpu_computing.html (дата обращения 13.12.2010).
2. NVIDIA CUDA C SDK Code. http://developer.download.nvidia.com/compute/cuda/sdk/website/samples.html (дата обращения 13.12.2010).
3. Карпенко А.П., Селиверстов Е.Ю. Глобальная безусловная оптимизации роем частиц на графических процессорах архитектуры CUDA // Наука и образование: электронное научно-техническое издание, 2010, 4. http://technomag.edu.ru/doc/142202.html (дата обращения 13.12.2010).
4. Штойер Р. Многокритериальная оптимизация. Теория, вычисления и приложения. М., Радио и связь, 1992. 504 с.
5. Карпенко А.П., Селиверстов Е.Ю. Глобальная оптимизация методом роя частиц. Обзор // Информационные технологии. 2010. № 2. С. 25-34.
6. Hu X., Eberhart R. Multiobjective optimization using dynamic neighborhood particle swarm optimization // World Congress on Computational Inelligence. Proceeding: 2002. P. 1677–1681.
7. Srinivas N., Deb K. Multiobjective optimization using nondominated sorting in genetic algorithms // Evolutionary Computation. 1994. vol. 2. P. 221–248.
8. Антух А.Э., Семенихин А.С., Хасанова Р.В. Построение множества Парето методом роя частиц на графических процессорах архитектуры CUDA // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи, Труды международной суперкомпьютерной конференции (21-26 сентября 2009 г., г. Новороссийск). М.: Изд-во МГУ, 2010. С.274-280.
9. Фролов В. Введение в технологию CUDA . http://cgm.computergraphics.ru/issues/issue16/cuda (дата обращения 13.12.2010).