ПОСТРОЕНИЕ МНОЖЕСТВА ПАРЕТО МЕТОДОМ РОЯ ЧАСТИЦ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ АРХИТЕКТУРЫ CUDA

А.Э. Антух, А.П. Карпенко, А.С. Семенихин, Р.В. Хасанова

Аннотация. В настоящее время при решении задач оптимизации все более широкое распространение получают стохастические поведенческие методы [1]. Одним из таких методов является метод роя частиц (Particle Swarm Optimization, PSO), основанный на закономерностях социального поведения [2, 3]. Достаточно новым является применение метода PSO в задаче многокритериальной оптимизации (Multi-Objective Swarm Optimization, MOPSO) [1]. В работе рассматривается применение этого метода для приближенного построения множества Парето в указанной задаче. Эффективность последовательной реализации MOPSO исследована в работе [4]. В данной работе рассматривается реализация метода на NVidia графических процессорах [5].

Постановка задачи. Совокупность частных критериев оптимальности $\phi_k(X), \mathrm{k}\in[1:\mathrm{s}]$ образует векторный критерий оптимальности $\Phi(X)$, где $X=(x_1,x_2,\ldots,x_n)$ - вектор варьируемых параметров. Ставится задача минимизации каждого из частных критериев в одной и той же области допустимых значений вектора варьируемых параметров $D_X\in R^n$. Запишем задачу многокритериальной оптимизации в виде

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

Положим, что частные критерии оптимальности тем или иным образом нормализованы.

Векторный критерий оптимальности $\Phi(X)$ выполняет отображение множества D_X в некоторую область $D_\Phi\subset\{\Phi\}$, где $\{\Phi\}$ - пространство критериев. Будем говорить, что вектор $X^1\in D_X$ предпочтительнее вектора $X^2\in D_X$, и писать $X^1>X^2$, если среди равенств и неравенств $\phi_k(X^1)\le\phi_k(X^2), \mathrm{k}\in[1:\mathrm{s}]$ имеется хотя бы одно строгое неравенство. Аналогично, будем говорить, что векторный критерий оптимальности $\Phi(X^1)\in D_\Phi$ доминирует векторный критерий оптимальности $\Phi(X^2)\in D_\Phi$, и писать $\Phi(X^1)>\Phi(X^2)$, если $X^1>X^2$ [6].

Во множестве D_Φ выделим подмножество D_Φ^* точек, для которых в этом подмножестве нет точек, их доминирующих. Подмножество D_Φ^* называется фронтом Парето, а подмножество $D_X^*\in D_X$, соответствующее множеству D_Φ^* - множеством Парето (переговорным множеством, областью компромисса).

Ставится задача приближенного построения множества Парето (а, тем самым, и фронта Парето) в задаче многокритериальной оптимизации (1).

Канонический метод роя частиц. Множество частиц (популяцию) обозначим $P=\{P_i, i\in[1:M]\}$, где M – количество частиц в рое (размер популяции). В дискретный момент времени $t\in[0:T]$ координаты частицы P_i определяются вектором $X_{i,t}=(x_{i,t,1},x_{i,t,2},\ldots,x_{i,t,n})$, а ее скорость - вектором $V_{i,t}=(v_{i,t,1},v_{i,t,2},\ldots,v_{i,t,n})$. Здесь T - количество итераций. Начальные координаты и скорости частицы P_i равны $X_{i,0}=X_i^0$, $V_{i,0}=V_i^0$ соответственно, где X_i^0 - $(n\times1)$ -вектор случайных чисел, а V_i^0 - аналогичный нулевой вектор. Состояние роя частиц P в момент времени t обозначим $S_t=\{X_t,V_t\}$, где $X_t=(X_{1,t},X_{2,t},\ldots,X_{M,t})$, $V_t=(V_{1,t},V_{2,t},\ldots,V_{M,t})$.

Итерации в каноническом методе PSO выполняются по следующей схеме:

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

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

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 то же время поиска значением целевой функции $\Phi(X)$; α,β,γ - свободные параметры алгоритма. Важнейшее в методе PSO понятие соседства частиц зависит

от используемой топологии соседства и определено, например, в работе [2]. В процессе итераций вектор $X^b_{i,t}$ образует так называемый собственный путь (private guide) частицы P_i, а вектор $X_{g,t}$ - локальный путь (local guide) этой частицы.

Пересчет координат частиц по формулам (2), (3) может происходить по синхронной схеме (обновление координат частиц выполняется только после определения текущих скоростей всех M частиц) или по асинхронной схеме (расчет новых координат части производится до завершения указанных вычислений).

Свободный параметр α в формуле (2) определяет «инерционные» свойства частиц (при $\alpha<1$ движение частиц замедляется). Рекомендуемое значение параметра α равно 0,7298 [2]. В процессе оптимизации может быть эффективным постепенное уменьшение коэффициента α от 0.9 до 0.4. При этом большие значения параметра обеспечивают широкий обзор пространства поиска, а малые – точную локализацию минимума целевой функции. Рекомендуемые значения свободных параметров β,γ равны 1,49618 [2].

Второй компонент в формуле (2) называется «когнитивным» компонентом (по социальной аналогии) и формализует тенденцию частицы вернуться в положение с минимальным значением целевой функции. Третий компонент в формуле (2) называется «социальным» компонентом. Компонент отражает влияние на данную частицу ее соседей.

Метод многокритериальной оптимизации роем частиц. Важной составной частью метода MOPSO является определение глобально лучшей частицы (в смысле формулы (1)) для каждой частицы в популяции. В многокритериальной задаче глобально лучшая частица отыскивается на множестве Парето. С этой целью в методе MOPSO используется архив частиц A, в котором хранятся координаты недоминируемых частиц $X_{i,t}$. Схема метода MOPSO имеет вид, представленный на рисунке 1.

Шаг 1. Инициализация
$t:=0$
Инициализация популяции
for $i=1$ to M
$X_{i,t}:=X_i^0$; $V_{i,t}:=V_i^0$; $X^b_{t,0}:=X_i^0$
end
Инициализация архива частиц
$A_t:=\{\}$
Шаг 2. Обновление архива частиц
$A_{t+1}:=Update(S_t,A_t)$
Шаг 3. Вычисление вектора глобально лучших частиц
for $i=1$ to M
$X^g_{i,t}:=FindGlobalBest(A_{t+1},X^b_{i,t})$;
for $j=1$ to n
$V_{i,t+1}:=\alpha V_{i,t}+U_1[0,\beta]\otimes\left(X^b_{i,t}-X_{i,t}\right)+U_2[0,\gamma]\otimes\left(X^g_{i,t}-X_{i,t}\right)$;
$X_{i,t+1}:=X_{i,t}+V_{i,t+1}$
end;
if ($X_{i,t}<X^b_{i,t}$) then $X^b_{i,t}:=X_{i,t}$;
end
Шаг 4. Проверка критерия останова итераций
$t:=t+1$;
if $t\le T$ then go to шаг 2

Рис.1. Схема метода MOPSO

В этой схеме функция Update выполняет сравнение координат частиц из архива A_t с координатами частиц, полученными на текущей итерации t. Если некоторая частица текущего поколения P_i доминирует частицу P_j из архива, то координаты частицы P_i заменяют в архиве координаты частицы P_j. Заметим, что на первой итерации, когда $t=0$ и архив пуст, функция Update добавляет в архив начальные состояния всех частиц, которые не доминируют друг друга.

Выбор глобально лучшей частицы осуществляет функция FindGlobalBest. Существует несколько способов реализации этой функции. В данной работе используется метод «меняющихся» соседей» Хью и

Эберхарта [7]. Рассмотрим суть этого метода на примере задачи двухкритериальной оптимизации. Поиск глобально лучшей частицы для каждой частицы популяции осуществляется в этом случае следующим образом: сначала вычисляем расстояние от частицы P_i до других частиц архива 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 $\phi_2(X)$, находим для частицы P_i из числа этих соседей наилучшую частицу, которая и принимается в качестве глобально лучшей частицы для частицы P_i.

Основные особенности архитектуры и программирования ГПУ. Основу ГПУ компании nVidia, таких как видеокарта nVidia GeForce 8 и выше, составляет массив, включающий в себя от четырех до тридцати двух SIMD-мультипроцессоров. Мультипроцессор состоит из восьми суперскалярных потоковых процессоров, каждый из которых имеет G_{reg}=16384 регистров. Мультипроцессор включает в себя независимую кэш-память команд и аналогичную память данных (констант), планировщик потоков и модуль памяти, разделяемой между всеми восемью потоковыми процессорами. Емкость разделяемой памяти составляет G_{shm}=16 КБ. Помимо собственно ГПУ, видеокарта содержит также модули глобальной оперативной памяти емкостью до 1 ГБ.

Основными вычислительными устройствами ГПУ являются RISC потоковые процессоры, которые при вычислениях исполняют одну и ту же вычислительную программу (шейдер) над разными исходными данными, находящимися в их регистрах общего назначения. Совокупность исходных данных, которые обрабатываются шейдером, удобно представлять в виде некоторой целочисленной решетки $\Omega=\{\varpi_i, i\in[1:M]\}$, узлы которой ϖ_i ассоциируются с этими данными [8]. Система управления ГПУ осуществляет балансировку загрузки ГПУ путем распределения узлов ϖ_i по потоковым процессорам. Для достижения высокой эффективности решения задачи на ГПУ число узлов решетки должно многократно превосходить число потоковых процессоров. Исполнение шейдера на одном из потоковых процессоров ГПУ называется потоком. Коммуникация потоков может осуществляться через разделяемую память с применением барьерной синхронизации.

Известной проблемой современных процессоров является дисбаланс между скоростью работы процессора и скоростью обмена с памятью. Для ГПУ эта проблема стоит еще более остро, поскольку задержка доступа потокового процессора к глобальной памяти может составлять до 300 тактов. Для того чтобы снизить влияние латентности доступа к глобальной памяти, в ГПУ используется следующий механизм. Потоки одного блока объединяются в некоторое число совокупностей, называемых варпами (warps), каждая из которых включает в себя до 32 потоков. Планировщик потоков планирует исполнение на мультипроцессоре варпов. При блокировании по доступу к памяти потоков одного из варпов, планировщик потоков быстро переключает мультипроцессор на исполнение следующего варпа. Важно, что при этом переключение контекста занимает только один такт.

Отображение MOPSO на архитектуру ГПУ. В простейшем случае отображение MOPSO на архитектуру ГПУ может быть выполнено по следующей схеме:

1. сопоставляем узлу ϖ_i решетки Ω частицу P_i (точнее, ее состояние S_i); $i\in[1:M]$;
2. шейдеру (обозначим его Sh_1) ставим в соответствие задачу вычисления по формулам (2), (3) состояний частицы P_i в моменты времени $t\in[0:T]$.

Из схемы метода MOPSO, представленной на рисунке 1, следует, что при использовании шейдера Sh_1 на каждой итерации требуется обновление архива частиц. Эта операции является последовательной и должна выполняться на центральном процессоре системы. Поэтому шейдер Sh_1 не может обеспечить высокое ускорение.

При высокой размерности пространства поиска R^n целесообразной может быть следующая простая модификация рассмотренной схемы отображения: узлу ϖ_i решетки Ω сопоставляем один из компонентов вектора координат и вектора скоростей частицы P_i; шейдеру (Sh_2) ставим в соответствие задачу вычисления по формулам (2), (3) состояния частицы P_i по соответствующему измерению. Отмеченный выше недостаток метода MOPSO при этом, очевидно, сохраняется.

Повысить эффективность метода MOPSO можно, изменив его схему следующим образом. Будем производить обновление архива не после каждой итерации, а после некоторого фиксированного числа итераций q. Обозначим соответствующий шейдер Sh_3.

Кардинально повысить эффективность параллельных вычислений можно путем комбинации метода MOPSO не с каноническим методом роя частиц, а с многороевым методом на основе островной модели параллелизма [9].

Последовательная реализация. Исследование эффективности метода MOPSO при его последовательной реализации выполнено для простой двухмерной двухкритериальной тестовой задачи, для которой известно точное решение, что позволяет произвести оценку точности полученной аппроксимации [4]:

1. множество допустимых значений вектора варьируемых параметров

$$D_X = \{X \mid 0 \le x_i \le 1, i \in [1,2]\} \subset R^2; \quad (4)$$

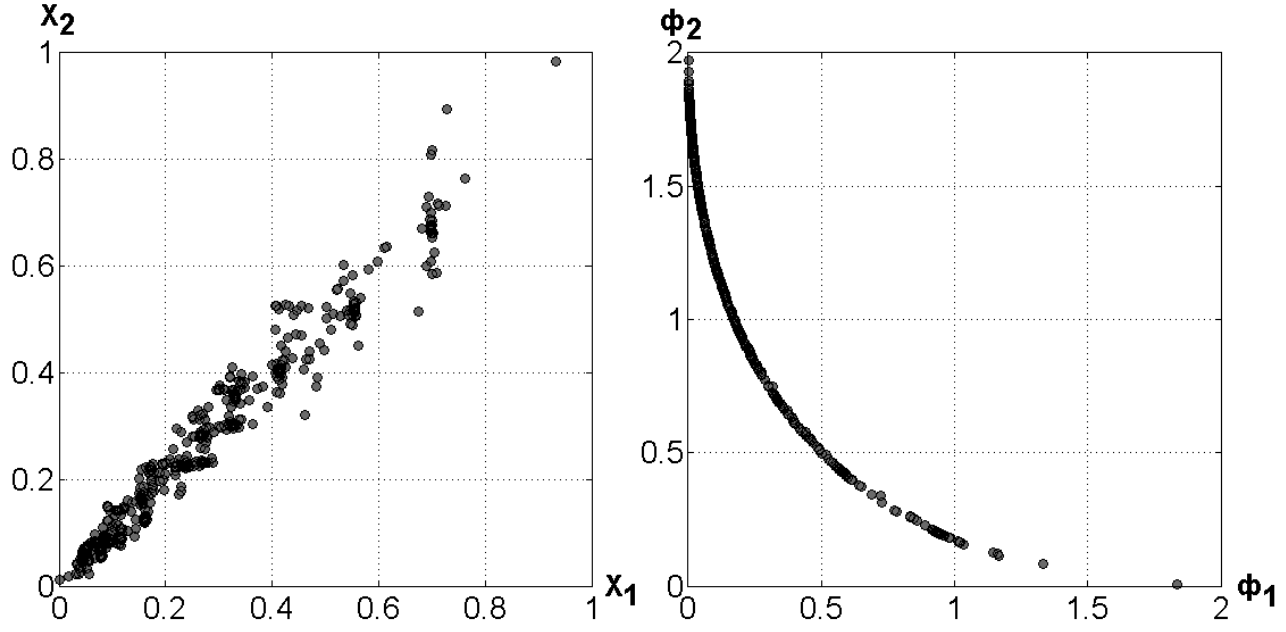
2. частные критерии оптимальности

$$\begin{cases} \varphi_1(X) = x_1^2 + x_2^2, \\ \varphi_2(X) = (x_1 - 1)^2 + (x_2 - 1)^2 \end{cases} \quad (5)$$

Использованы следующие значения свободных параметров метода: размер популяции $M = 100$; число итераций $T = 500$; инерциальный коэффициент $\alpha = 0{,}7238$; когнитивный коэффициент $\beta = 1{,}6$; социальный коэффициент $\gamma = 1{,}6$;

Эффективность метода MOPSO в указанных условиях иллюстрируют рисунки 2 - 4. Отметим, что точное множество Парето для задачи (4), (5) представляет собой отрезок прямой, координаты концов которого равны (0, 0), (1, 1); точный фронт Парето – дуга окружности, проходящей через точки с координатами (0, 2), (2, 0), (0,5, 0,5). Рисунок 3 показывает, что метод сходится уже на первых 100-200 итерациях, после чего имеет место стагнация итерационного процесса. Здесь $\bar{\varepsilon}$ - средняя ошибка аппроксимации, $\sigma(\varepsilon)$ - среднее квадратичное отклонение этой ошибки. Рисунок 4 иллюстрирует динамику изменения объема архива $|A|$ в процессе итераций, а также время выполнения одной итерации с учетом времени на обновление архива τ. Из рисунка следует, что объем архива практически перестает изменяться уже после ~200 итераций. Важность этого результата состоит в том, что на его основании в качестве критерия окончания итераций можно предложить использование более экономного, чем фиксированное количество итераций, критерия: в течение t_Δ итераций размер архива увеличился не более, чем на величину $|A_\Delta|$. Здесь t_Δ, $|A_\Delta|$ - заданные константы.

Заметим, что в рассматриваемом варианте алгоритма, реализующего метод MOPSO, использовался архив A, не имеющий ограничения на его размер. Для практических задач многокритериальной оптимизации размер архива приходится ограничивать. Таким образом появляется возможность использовать еще один критерий окончания итераций – полное заполнение архива.



а) б)

Рис. 2. Аппроксимация множества Парето (а) и фронта Парето (б) для задачи (4), (5): последовательная реализация

Параллельная реализация. В этом случае в качестве тестовой использовалась двухкритериальная 16-мерная задача:

1. множество допустимых значений вектора варьируемых параметров

$$D_X = \{X \mid 0 \le x_i \le 1, i \in [1{:}16]\} \subset R^{16}; \quad (6)$$

2. частные критерии оптимальности

$$\begin{cases} \varphi_1(X)=\sum_{i=1}^{16} x_i^2, \\ \varphi_2(X)=\sum_{i=1}^{16} \left(x_i-1\right)^2 \end{cases} \tag{7}$$

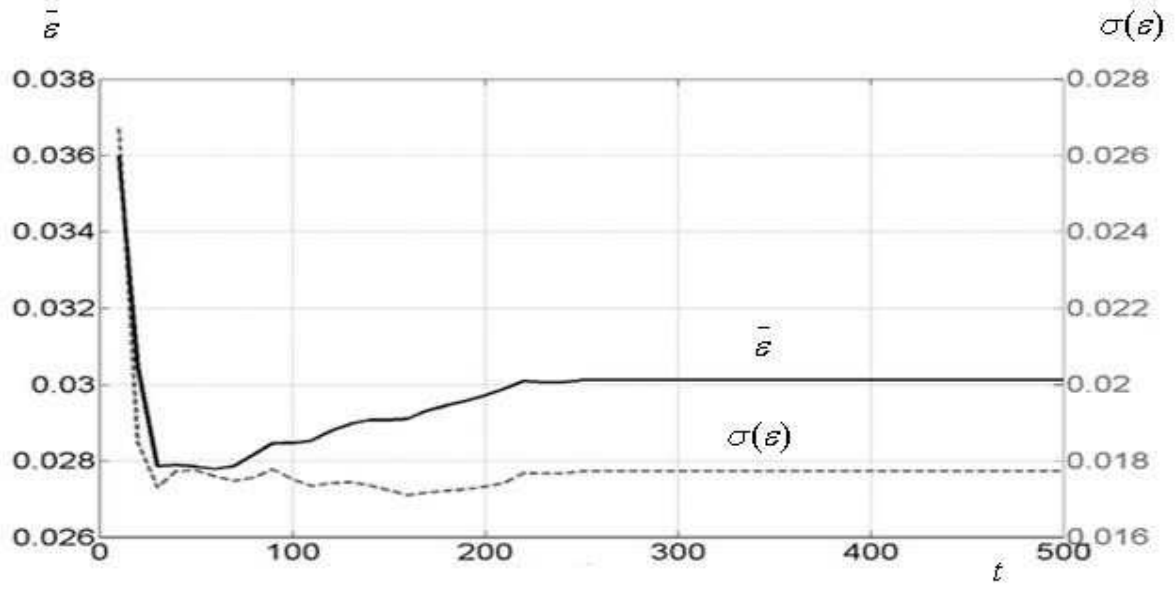


Рис. 3. Точность аппроксимации множества Парето

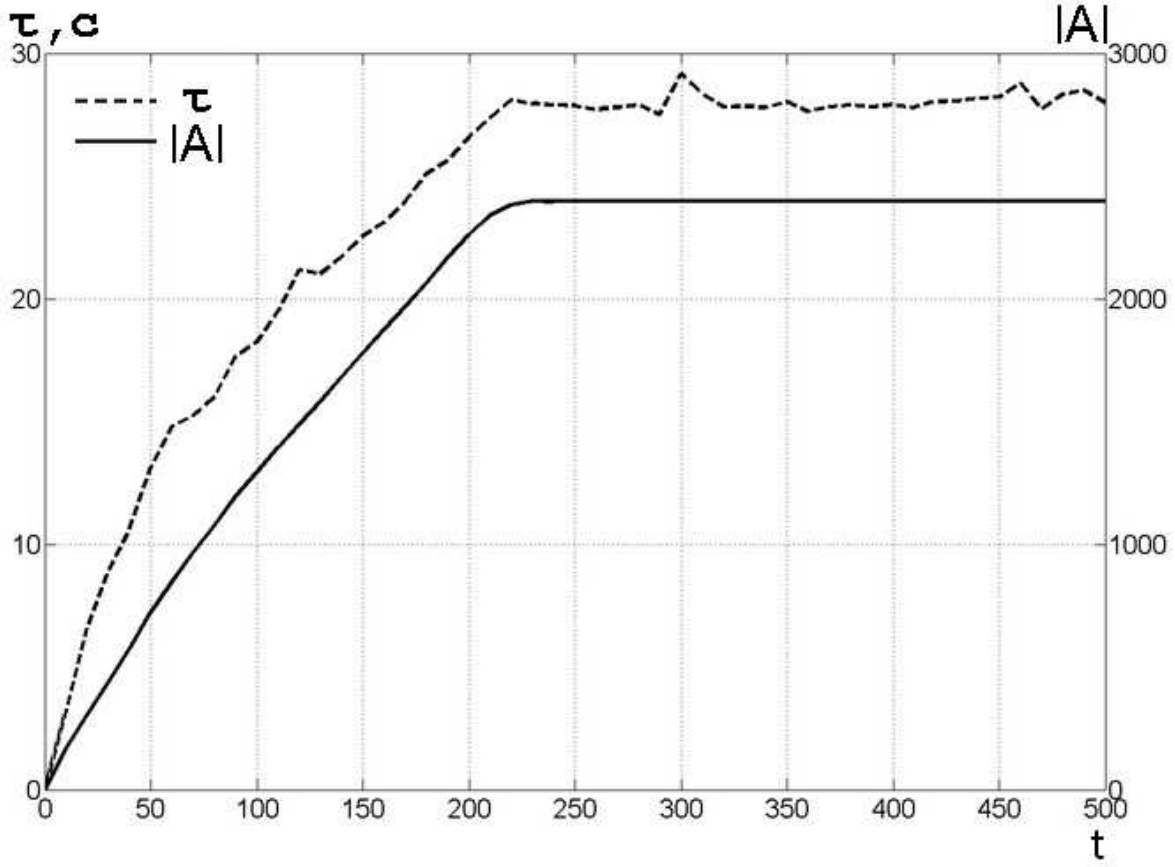


Рис. 4. К оценке производительности алгоритма MOPSO

Использованы следующие значения свободных параметров метода: размер популяции $M=256$; максимально допустимое число итераций $T=1000$; допустимое изменение объема архива (для определения момента останова) $|A_{\Delta}|=0$; допустимое число итераций (для определения момента останова) $t_{\Delta}=50$; инерциальный коэффициент $\alpha=0{,}7238$; когнитивный коэффициент $\beta=1{,}6$; социальный коэффициент $\gamma=1{,}6$;

Реализован шейдер Sh_1. Реализация выполнена на ГПУ NVidia GeForce 8500GT, которое содержит два мультипроцессора, каждый из которых включает в себя по 16 потоковых процессоров. В качестве центрального процессора использовался процессор Intel Pentium Dual E2160|1,8Гц/1ГБ.

Результаты решения задачи (6), (7) иллюстрирует рисунок 5, на котором приведена полученная аппроксимация фронта Парето. Ускорение, которое обеспечивает шейдер Sh_1, иллюстрирует рисунок 6, где τ_1 - время последовательного решения задачи на центральном процессоре системы, τ_2 - время параллельного решения задачи на ГПУ.

Рисунок 6 показывает, что параллельная реализация метода MOPSO с помощью шейдера Sh_1 обеспечивает всего лишь примерно двукратное ускорение. Как отмечалось выше, причиной этого является необходимость выполнения на каждой итерации последовательной операции, заключающейся в обновлении архива частиц. Для обеспечения более высокого ускорения необходимо использовать другой шейдер, построенный на основе многороевого метода MOPSO и островной модели параллелизма.

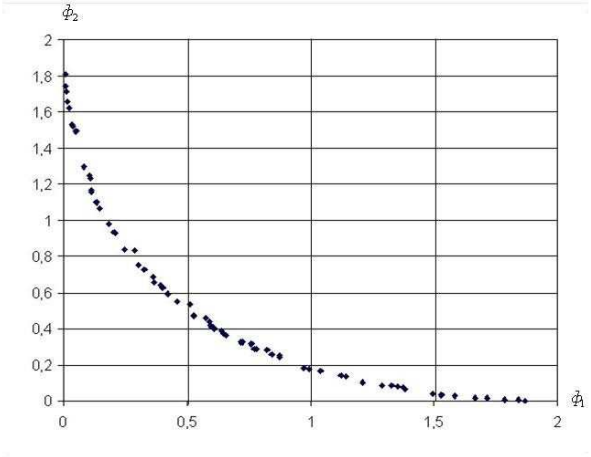


Рис. 5. Аппроксимация фронта Парето для задачи (6), (7): шейдер Sh_1

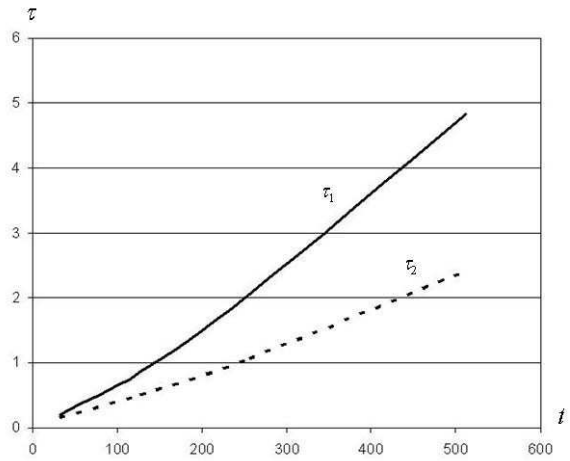


Рис. 6. К ускорению параллельной реализации метода MOPSO: шейдер Sh_1

Заключение. В работе рассмотрен метод MOPSO для приближенного построения множества Парето в задаче многокритериальной оптимизации с помощью роя частиц. Приведено несколько вариантов отображения этого метода на графические процессорные устройства с архитектурой CUDA. На ГПУ NVidia GeForce 8500GT реализовано одно из предложенных отображений, а также исследована его эффективность. Результаты исследования показывают, что метод MOPSO, будучи относительно простым (как в математическом плане, так и в плане реализации), обеспечивает решение задачи с приемлемой точностью. По причинам, которые указаны в работе, рассмотренная реализация метода обеспечивает низкое ускорение.

В развитии работы планируется реализация и исследование эффективности нескольких модификаций метода MOPSO, в частности многороевого метода MOPSO на основе островной модели параллелизма. Кроме того, предполагается реализация метода и его модификаций на вычислительной системе, содержащей несколько ГПУ.

Авторы выражают благодарность Е.Ю. Селиверстову за плодотворные обсуждения постановки задачи, методов ее решения, а также за помощь в программировании ГПУ.

ЛИТЕРАТУРА:

1. S. Mostaghim, J. Teich. Strategies for Finding Good Local Guides in Multi-Objective Particle Swarm Optimization (MOPSO) // Swarm Intelligence Symposium: Proceeding, 2003. - pp. 26–33.
2. А.П. Карпенко, Е.Ю. Селиверстов. Глобальная оптимизация методом роя частиц. Обзор // Информационные технологии, 2010, № 2, с. 25-34.

3. С.А. Субботин, Ан.А. Олейник, Ал.А. Олейник. PSO-метод, «Интеллектуальные мультиагентные методы (Swarm Intelligence)», 2006, №3, с. 55-70.
4. А.Э. Антух, А.С. Семенихин, Р.В. Хасанова. Исследование эффективности метода роя частиц в задаче приближенного построения множества Парето // XII молодежная международная научно-техническая конференция «Наукоемкие технологии и интеллектуальные системы 2010». 12 апреля 2010 г., Москва, МГТУ им. Н.Э. Баумана, с.34-37.
5. NVidia Tesla C1060. URL: (http://www.nvidia.com/object/product_tesla_c1060_us.html).
6. И.Г. Черноруцкий. Методы принятия решений. –СПб.: БХВ-Петербург, 2005.-416 с.
7. X. Hu, Eberhart R. Multiobjective optimization using dynamic neighborhood particle swarm optimization // World Congress on Computational Inelligence: Proceeding, 2002.- pp. 1677–1681.
8. А. Адинец, Вл.В Воеводин. Графический вызов суперкомпьютерам // Открытые системы. 2008, № 4, с. 35–41.
9. А.П. Карпенко, Е.Ю. Селиверстов. 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).