Решение задач глобальной оптимизации на графических ускорителях*

К.А. Баркалов, И.Г. Лебедев

Нижегородский государственный университет им. Н.И. Лобачевского

В работе развивается подход к решению задач глобальной оптимизации, использующий схему вложенной оптимизации. Новым элементом является использование разных алгоритмов на разных уровнях вложенности: сложный последовательный алгоритм (на CPU) – на верхнем уровне; простой параллельный алгоритм (на GPU) - на нижнем уровне. Данная схема вычислений реализована в параллельном решателе ExaMin. Приведены результаты вычислительных экспериментов, демонстрирующие ускорение при решении серии тестовых задач.

Ключевые слова: глобальная оптимизация, многоэкстремальные функции, редукция размерности, параллельные алгоритмы, графические ускорители

1. Введение

Задача многомерной многоэкстремальной оптимизации может быть определена как проблема поиска наименьшего значения действительной функции $\varphi(y)$

$$\begin{aligned} &\varphi(y^*) = \min\{\varphi(y) : y \in D\}, \\ &D = \{y \in R^N : a_i \le y_i \le b_i, 1 \le i \le N\}, \end{aligned} \quad (1)$$

где $a, b \in R^N$ есть заданные векторы.

Численное решение задачи (1) сводится к построению оценки $y_k^* \in D$, отвечающей некоторому понятию близости к точке y^* (например, $\| y^* - y_k^* \| \le \varepsilon$, где $\varepsilon > 0$ есть заданная точность) на основе конечного числа k вычислений значений оптимизируемой функции. Относительно класса рассматриваемых задач предполагается выполнение двух важных условий.

Во-первых, предполагается, что оптимизируемая функция $\varphi(y)$ может быть задана не аналитически, в виде формулы, а алгоритмически, как результат работы некоторой подпрограммы или библиотеки.

Во-вторых, будем предполагать, что $\varphi(y)$ удовлетворяет условию Липшица

$$|\varphi(y_1) - \varphi(y_2)| \le L\|y_1 - y_2\|, \ y_1, y_2 \in D, \ 0 < L < \infty, \quad (2)$$

что соответствует ограниченности изменения значений функции при ограниченной вариации аргумента. Это предположение можно интерпретировать (применительно к прикладным задачам) как отражение ограниченности мощностей, порождающих изменения в моделируемой системе.

Задачи многоэкстремальной оптимизации имеют существенно более высокую трудоемкость решения по сравнению с другими типами оптимизационных задач, т.к. глобальный оптимум является интегральной характеристикой решаемой задачи и требует исследования всей области поиска. Как результат, поиск глобального оптимума сводится к построению некоторого покрытия (сетки) в области параметров, и выборе наилучшего значения функции на данной сетке. Вычислительные затраты на решение задачи растут экспоненциально с ростом размерности.

В ННГУ им. Н.И. Лобачевского под руководством проф. Р.Г. Стронгина разработан эффективный подход к решению задач глобальной оптимизации [1–7]. В рамках данного подхода решение многомерных задач сводится к решению серии вложенных задач меньшей размерно-

* Исследование выполнено при поддержке Российского научного фонда, проект № 15-11-30022.

сти. Для эффективного решения быстро вычисляемых задач предлагается полностью перенести решение вложенной задачи на графический ускоритель.

2. Базовый параллельный алгоритм глобального поиска

В качестве базовой задачи мы будет рассматривать одномерную задачу многоэкстремальной оптимизации $\varphi^* = \varphi(x^*) = \min\{\varphi(x) : x \in [0,1]\}$, в которой целевая функция $\varphi(y)$ удовлетворяет условию Липшица. Дадим детальное описание параллельного алгоритма глобального поиска (ПАГП), применяемого к ее решению.

Пусть в нашем распоряжении имеется $p \geq 1$ вычислительных элементов. Тогда на данной итерации можно провести одновременно p испытаний. Тогда общее число испытаний, выполненных после n параллельных итераций, составит $k = pn$.

Предположим, что выполнено $n > 1$ итераций метода (в качестве точек $x^1,..., x^p$ первой итерации выбираются произвольные различные точки отрезка [0,1]). Тогда точки $x^{k+1},..., x^{k+p}$ текущей $(n+1)$-ой итерации определяются по следующим правилам.

Правило 1. Перенумеровать точки множества $X_k = \{x^1,..., x^k\} \cup \{0\} \cup \{1\}$ так что

$$0 = x_0 < x_1 < ... < x_{k+1} = 1.$$

Правило 2. Полагая $z_i = \varphi(x_i), 1 \leq i \leq k$, вычислить величины

$$\mu = \max_{1 \leq i \leq k} \frac{|z_i - z_{i-1}|}{\Delta_i}, \; M = \begin{cases} r\mu, \mu > 0, \\ 1, \mu = 0, \end{cases} \tag{3}$$

где $r > 1$ является заданным параметром метода (параметр надежности), а $\Delta_i = x_i - x_{i-1}$.

Правило 3. Для каждого интервала $(x_{i-1}, x_i), 1 \leq i \leq k+1$, вычислить характеристику в соответствии с формулами

$$R(1) = 2\Delta_1 - 4\frac{z_1}{M}, \quad R(k+1) = 2\Delta_{k+1} - 4\frac{z_k}{M}, \tag{4}$$

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{M^2 \Delta_i} - 2\frac{z_i + z_{i-1}}{M}, \quad 1 < i < k+1, \tag{5}$$

Правило 4. Характеристики $R(i), 1 \leq i \leq k+1$, упорядочить в порядке убывания

$$R(t_1) \geq R(t_2) \geq \ldots \geq R(t_{k-1}) \geq R(t_{k+1}) \tag{6}$$

и выбрать p наибольших характеристик с номерами интервалов $t_j, 1 \leq j \leq p$.

Правило 5. Провести новые испытания в точках $x^{k+j}, 1 \leq j \leq p$, вычисленных по формулам

$$\begin{aligned} x^{k+j} &= \frac{x_{t_j} + x_{t_j-1}}{2}, \; t_j = 1, \; t_j = k+1, \\ x^{k+j} &= \frac{x_{t_j} + x_{t_j-1}}{2} - \frac{z_{t_j} - z_{t_j-1}}{2M}, \; 1 < t_j < k+1. \end{aligned} \tag{7}$$

Алгоритм прекращает работу, если выполняется условие $\Delta_{t_j} \leq \varepsilon$ хотя бы для одного номера $t_j, 1 \leq j \leq p$; здесь $\varepsilon > 0$ есть заданная точность. В качестве оценки глобально-оптимального решения задачи (1) выбираются значения

$$\varphi_k^* = \min_{1 \leq i \leq k} \varphi(x^i), \; x_k^* = \arg \min_{1 \leq i \leq k} \varphi(x^i).$$

Данный способ организации параллельных вычислений имеет следующее обоснование [10, 11]. Используемые в алгоритме характеристики интервалов (5) могут рассматриваться как некоторые меры вероятности локализации в данных интервалах точки глобального минимума. Неравенства (6) упорядочивают интервалы по их характеристикам, и испытания проводятся

параллельно в первых *p* интервалах, имеющих наибольшие вероятности. Различные модификации данного алгоритма и соответствующая теория сходимости представлены в [11].

3. Редукция размерности

3.1 Редукция размерности с использованием кривых Пеано

Для снижения сложности алгоритмов глобальной оптимизации, формирующих неравномерное покрытие области поиска, широко используются различные схемы редукции размерности, которые позволяют свести решение многомерных оптимизационных задач к семейству задач одномерной оптимизации.

Первым из рассматриваемых способов редукции размерности является использование кривой Пеано y(x), однозначно отображающей отрезок вещественной оси [0,1] на n-мерный куб

$$\left\{y \in R^N : -2^{-1} \le y_i \le 2^{-1}, 1 \le i \le N\right\} = \left\{y(x) : 0 \le x \le 1\right\}.$$

Вопросы численного построения отображений типа кривой Пеано и соответствующая теория подробно рассмотрены в [10]. Здесь же отметим, что численно построенная развертка является приближением к теоретической кривой Пеано с точностью порядка 2^{-m}, где m – параметр построения развертки.

Использование подобного рода отображений позволяет свести многомерную задачу (1) к одномерной задаче

$$\varphi(y^*) = \varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0,1]\}.$$

Важным свойством является сохранение ограниченности относительных разностей функции: если функция $\varphi(y)$ в области D удовлетворяла условию Липшица (2) с константой L, то функция $\varphi(y(x))$на интервале [0,1] будет удовлетворять равномерному условию Гельдера

$$\left|\varphi(y(x_1)) - \varphi(y(x_2))\right| \le H\left|x_1 - x_2\right|^{1/N},\ x_1, x_2 \in [0,1], \tag{8}$$

где константа Гельдера H связана с константой Липшица L соотношением

$$H = 4Ld\sqrt{N},\ d = \max\{b_i - a_i : 1 \le i \le N\}.$$

Соотношение (8) позволяет модифицировать приведенный в разделе 2 алгоритм решения одномерных задач для решения многомерных задач, редуцированных к одномерным. Для этого длины интервалов Δ_i, участвующие в правилах (3)–(5) алгоритма, заменяются на длины в новой метрике $\Delta_i = (x_i - x_{i-1})^{1/N}$, а вместо формулы (7) вводится выражение

$$x^{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2} - \mathrm{sign}(z_{t_j} - z_{t_j-1})\frac{1}{2r}\left[\frac{|z_{t_j} - z_{t_j-1}|}{\mu}\right]^N,\ 1 < t_j < k+1.$$

3.2 Рекурсивная схема редукции размерности

Схема рекурсивной оптимизации основана на известном (см. [12]) соотношении

$$\min\{\varphi(y) : y \in D\} = \min_{a_1 \le y_1 \le b_1} \min_{a_2 \le y_2 \le b_2} \ldots \min_{a_N \le y_N \le b_N} \varphi(y), \tag{9}$$

которое позволяет заменить решение многомерной задачи (1) решением семейства одномерных подзадач, рекурсивно связанных между собой.

Введем в рассмотрение множество функций

$$\varphi_N(y_1, \ldots, y_N) = \varphi(y_1, \ldots, y_N), \tag{10}$$

$$\varphi_i(y_1, \ldots, y_i) = \min_{a_{i+1} \le y_{i+1} \le b_{i+1}} \varphi_{i+1}(y_1, \ldots, y_i, y_{i+1}),\ 1 \le i \le N-1. \tag{11}$$

Тогда, в соответствии с соотношением (9), решение исходной задачи (1) сводится к решению одномерной задачи

$$\varphi_1(y_1^*) = \min\{\varphi_1(y_1) : y_1 \in [a_1, b_1]\}. \quad (12)$$

Однако при этом каждое вычисление значения одномерной функции $\varphi_1(y_1)$ в некоторой фиксированной точке предполагает решение одномерной задачи минимизации

$$\varphi_2(y_1, y_2^*) = \min\{\varphi_2(y_1, y_2) : y_2 \in [a_2, b_2]\},$$

и так далее до вычисления φ_N согласно (10).

Для изложенной выше рекурсивной схемы предложено обобщение (блочная рекурсивная схема), которое комбинирует использование разверток и рекурсивной схемы с целью эффективного распараллеливания вычислений.

Рассмотрим вектор *y* как вектор блочных переменных

$$y = (y_1, y_2, ..., y_N) = (u_1, u_2, ..., u_M),$$

где *i*-я блочная переменная u_i представляет собой вектор размерности N_i из последовательно взятых компонент вектора *y*, т.е. $u_1 = (y_1, y_2, ..., y_{N_1})$, $u_2 = (y_{N_1+1}, y_{N_1+2}, ..., y_{N_1+N_2})$,…, $u_2 = (y_{N-N_M+1}, y_{N-N_M+2}, ..., y_N)$, причем $N_1 + N_2 + ... + N_M = N$.

С использованием новых переменных основное соотношение многошаговой схемы (9) может быть переписано в виде

$$\min_{y \in D} \varphi(y) = \min_{u_1 \in D_1} \min_{u_2 \in D_2} ... \min_{u_M \in D_M} \varphi(y), \quad (13)$$

где подобласти $D_i, 1 \le i \le M$, являются проекциями исходной области поиска *D* на подпространства, соответствующие переменным $u_i, 1 \le i \le M$.

Формулы, определяющие способ решения задачи (1) на основе соотношений (13) в целом совпадают с рекурсивной схемой (10)–(12). Требуется лишь заменить исходные переменные $y_i, 1 \le i \le N$, на блочные переменные $u_i, 1 \le i \le M$.

При этом принципиальным отличием от исходной схемы является тот факт, что в блочной схеме вложенные подзадачи

$$\varphi_i(u_1, ..., u_i) = \min_{u_{i+1} \in D_{i+1}} \varphi_{i+1}(u_1, ..., u_i, u_{i+1}),\ 1 \le i \le M-1, \quad (14)$$

являются многомерными, и для их решения может быть применен способ редукции размерности на основе кривых Пеано.

Число векторов и количество компонент в каждом векторе являются параметрами блочной многошаговой схемы и могут быть использованы для формирования подзадач с нужными свойствами. Например, если $M = N$, т.е. $u_i = y_i, 1 \le i \le N$, то блочная схема идентична исходной; каждая из вложенных подзадач является одномерной. А если $M = 1$, т.е. $u = u_1 = y$, то решение задачи эквивалентно ее решению с использованием единственной развертки, отображающей [0,1] в *D*; вложенные подзадачи отсутствуют.

4. Организация параллельных вычислений

Для организации параллельных вычислений будем использовать небольшое (2-3) число уровней вложенности, при котором исходная задача большой размерности разбивается на 2-3 вложенные подзадачи меньшей размерности. Тогда, применяя в блочной рекурсивной схеме (13) для решения вложенных подзадач (14) параллельные характеристические методы глобальной оптимизации, мы получим параллельный алгоритм с широкой степенью вариативности. Например, можно варьировать количество процессоров на различных уровнях оптимизации (т.е. при решении подзадач по различным переменным u_i), применять различные параллельные методы поиска на разных уровнях и т.д.

На данный момент для быстрого решения задач на графическом ускорителе реализован алгоритм перебора, строящий равномерное покрытие области поиска. Пересылки данных от CPU к GPU будут минимальные: требуется лишь передать на GPU фиксированные координаты точки испытания, и получить обратно координаты и значения найденной точки глобального мини-

мума. Этот простой алгоритм позволяет оценить возможности данного подхода к решению многомерных задач глобальной оптимизации и применяется только для решения задач небольшой размерности (не более пяти).

Общая схема организации вычислений с использованием нескольких узлов кластера и нескольких GPU приведена на рис. 1; процессы параллельной программы будут образовывать дерево, соответствующее уровням вложенных подзадач. В соответствии с данной схемой вложенные подзадачи $\varphi_i(u_1,...,u_i) = \min_{u_{i+1} \in D_{i+1}} \varphi_{i+1}(u_1,...,u_i,u_{i+1})$ при i=1,…,M–2 решаются только с использованием CPU. Непосредственно в данных подзадачах вычислений значений оптимизируемой функции не происходит: вычисление значения функции $\varphi_i(u_1,...,u_i)$ – это решение задачи минимизации следующего уровня. Каждая подзадача решается в отдельном процессе; обмен вычисленными значениями организован с помощью MPI.

Подзадача последнего (M–1)-го уровня $\varphi_{M-1}(u_1,...,u_{M-1}) = \min_{u_M \in D_M} \varphi_M(u_1,...,u_M)$ отличается всех предыдущих подзадач – в ней происходит вычисление значений оптимизируемой функции, т.к. $\varphi_M(u_1,...,u_M) = \varphi(y_1,...,y_N)$. Данная подзадача целиком решается на GPU, она также выполняется в отдельном процессе.

Отметим также, что в случае невозможности эффективно реализовать процесс вычисления значения оптимизируемой функции на GPU на последнем уровне распараллеливания можно использовать ядра центрального процессора.

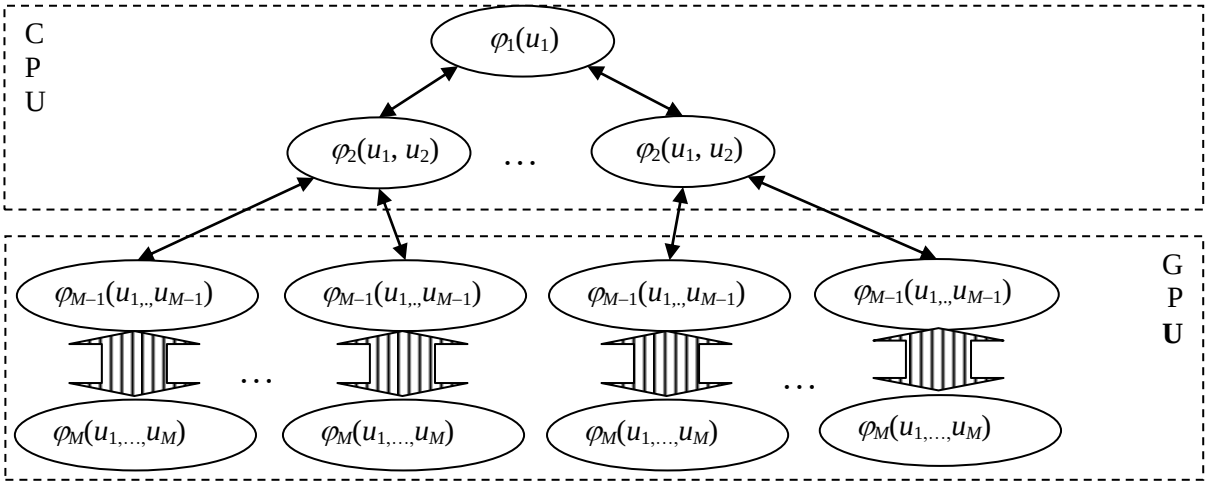

Рис. 1. Схема организации параллельных вычислений на кластере

5. Результаты вычислительных экспериментов

Вычислительные эксперименты проводились на одном из узлов вычислительного кластера ННГУ им. Н.И. Лобачевского. Узел кластера располагает 2-я процессорами Intel Xeon L5630 2.13 GHz, 24 Gb RAM и двумя видео картами NVIDIA Tesla X2070. Центральный процессор является 4-х ядерным.

Рассмотренные в разделе 3 методы и их модификации реализованы в решателе ExaMin, предназначенном для параллельного решения многомерных многоэкстремальных задач глобальной оптимизации, разрабатываемом в ННГУ им. Н.И. Лобачевского. Алгоритмическую основу решателя ExaMin составляют алгоритм глобального поиска и блочная многошаговая схема редукции размерности [16]. В работах [8, 9] описан GKLS-генератор, позволяющий порождать задачи многоэкстремальной оптимизации с заранее известными свойствами: количеством локальных минимумов, размерами их областей притяжения, точкой глобального минимума, значением функции в ней и т.п.

Согласно базовому параллельному алгоритму глобального поиска испытания проводятся параллельно на CPU, в работе [15] описана реализация данного подхода на GPU. В начале рассмотрим решение задач малой размерности с помощью ПАГП работающего в многопоточном режиме на центральном процессоре и графическом ускорителе. Численное сравнение проводи-

лось на классах функций Simple и Hard размерности 2 – 5 из [14]. Глобальный минимум y^* считался найденным, если алгоритм генерировал точку испытания y^k в δ-окрестности глобального минимума, т.е. $\|y^k - y^*\| \le \delta$. При этом размер окрестности выбирался (в соответствии с [14]) как $\delta = \|b-a\|\sqrt[N]{\Delta}$, здесь N – размерность решаемой задачи, a и b – границы области поиска D, параметр $\Delta=10^{-4}$ при N=2, $\Delta=10^{-6}$ при N=3, $\Delta=10^{-6}$ при N=4 и $\Delta=10^{-7}$ при N=5. При использовании метода АГП для класса Simple выбирался параметр r=4.5, для класса Hard – r=5.6; параметр построения кривой Пеано был фиксированный $m=10$. Максимально допустимое число итераций составляло K_{max} = 7 000 000. В таблице 1 приведено время работы в секундах и среднее число итераций для последовательного запуска.

Таблица 1. Среднее время и число итераций решения задачи на CPU

N	**Класс**	**Время решения**	**Число итераций**
2	hard	0.05	4731
	simple	0.02	2349
3	hard	0.08	5382
	simple	0.03	2129
4	hard	0.57	37410
	simple	0.19	12558
5	hard	3.84	247784
	simple	0.24	15538

В таблицах 2 и 3 приведено ускорение по времени, на CPU и GPU соответственно, относительно последовательного алгоритма (p=1). Значения приведены в зависимости от числа потоков p, для CPU от 2 до 16 и для GPU от 128 до 512. В запуске на GPU использовались два ускорителя.

Таблица 2. Ускорение по времени на CPU

N	**Класс**	***p*=2**	***p*=4**	***p*=8**	***p*=16**
2	hard	6,50	0,63	0,90	1,64
	simple	7,38	1,16	6,87	0,78
3	hard	1,01	1,12	1,33	1,29
	simple	1,10	1,33	1,36	1,43
4	hard	1,32	1,37	1,61	1,89
	simple	1,35	1,53	1,49	1,61
5	hard	1,20	1,29	2,22	1,75
	simple	0,68	0,99	0,62	1,71

Таблица 3. Ускорение по времени на GPU

N	**Класс**	***p*=128**	***p*=256**	***p*=512**
2	hard	4,97	6,11	4,36
	simple	4,60	3,37	2,27
3	hard	1,19	1,34	1,36
	simple	1,06	1,02	0,76
4	hard	1,38	1,51	1,53
	simple	1,12	1,60	1,59
5	hard	1,82	2,35	1,29
	simple	0,66	1,02	0,83

Результаты экспериментов показывают незначительное ускорение при решении задач с быстро вычисляемым критерием.

Далее рассмотрим решение задач перебором реализованным на графическом ускорителе. Максимально допустимое число итераций составляло K_{max} = 30 000 000. В таблице 4 приведено ускорение перебора и число решившихся задач перебором с шагом 0.01 при N=2 и N=3, для размерности 4 и 5 при шаге 0.01 требуется более K_{max} испытаний, поэтому вместо шага сетки 0.01 увеличен до 0.027 при N=4 и 0.064 при N=5, естественно что при малом шаге сетки решаются не все задачи. По завершению работы перебора происходит локальное уточнение методом Хука Дживса [13]. Ускорение приведены относительно последовательного АГП на центральном процессоре. Для вычислений использовались два ускорителя.

Таблица 4. Ускорение по времени и число решившихся задач

N	Класс	Ускорение	Решилось
2	hard	42.4	100
	simple	20.8	100
3	hard	4.4	100
	simple	1.8	100
4	hard	7.6	100
	simple	2.6	100
5	hard	45.7	77
	simple	3.0	88

Из таблицы видно для размерности 2, 3 и 4 за 30 000 000 испытаний решились все задачи при этом ускорение значительно лучше чем при использование параллельного вычисления только значений функции. Также поскольку время работы перебора зависит только от размерности, то ускорение на сложном классе больше чем на простом. При большом шаге сетки решаются не все задачи – локальный метод сходится к локальному минимуму.

Далее приведены результаты решения шестимерных и восьмимерных задач простого класса параметр r=4.5, $\Delta=10^{-8}$ при N=4 и $\Delta=10^{-9}$ при N=5, в соответствии с блочной рекурсивной схемой (13) было использовано два уровня подзадач с размерностями $N_1 = N_2 = 3$ для шестимерной и $N_1 = N_2 = 4$ для восьмимерной. Максимально допустимое число итераций составляло K_{max} = 1 000 000 на каждом уровне. Шаг сетки равен 0.1. По окончанию работы основного алгоритма применяется локальное уточнение. В таблице 5 приведено среднее время (в секундах) решения задачи в следующих режимах:

- на CPU с использованием последовательного алгоритма глобального поиска (столбец AGP);
- в параллельном режиме с использованием блочной рекурсивной схемы редукции размерности, на каждом уровне вложенности используется один процесс, вычисления производятся на CPU (столбец B-AGP);
- в гибридном режиме с использованием блочной рекурсивной схемы редукции размерности, на каждом уровне один процесс, на первом уровне вычисления производятся на центральном процессоре, на втором уровне используется перебор реализованный на GPU, вычисления производятся на одном узле кластера с двумя ускорителями (H-AGP).
- в гибридном режиме с использованием блочной рекурсивной схемы редукции размерности, на первом уровне вычисления производятся в один процесс на центральном процессоре одного узла кластера, на втором уровне используется перебор реализованный на GPU, использовалось четыре процесса, каждый запущен на отдельном узле кластера. Поскольку процессы второго уровня практически не используют центральный процессор на одном из узлов кластера было запущено два процесса (по одному первого и второго уровня), таким образом было задействовано четыре узла кластера (M-AGP).

В таблице 6 приведено ускорение относительно последовательного запуска. В скобках указано число не решившихся задач.

Таблица 5. Среднее время решения задачи большой размерности

N	AGP	B-AGP	H-AGP	M-AGP
6	53,5(20)	4,4	1,1	0,4
8	72,6(19)	78,3	10,7	3,1

Таблица 6. Ускорение по времени для решения задач большой размерности

N	B-AGP	H-AGP	M-AGP
6	12,1	48,4	133.7
8	0,9	6,8	23.4

Результаты экспериментов показывают значительное ускорение при использовании перебора для задач небольшой размерности. При использовании блочной рекурсивной схемы редукции размерности решаются все задачи.

6. Заключение

Результаты проведенных экспериментов на серии тестовых задач разной размерности с быстро вычисляемым критерием показывают, что предложенная блочная многошаговая схема редукции размерности в сочетании с простым алгоритмом (перебором по равномерной сетке в области параметров) эффективно реализуется на современных вычислительных системах.

Литература

1. В.П. Гергель. Об одном способе учета значений производных при минимизации многоэкстремальных функций //Журнал вычислительной математики и математической физики. 1996. Т. 36, № 6. С. 51–67.
2. Gergel V.P., Sergeyev Ya.D. Sequential and parallel algorithms for global minimizing functions with Lipschitzian derivatives // Computers and Mathematics with Applications 1999. Vol. 37 No. 4-5. P. 163–179.
3. Gergel V.P., Strongin R.G. Parallel computing for globally optimal decision making on cluster systems // Future Generation Computer Systems 2005. Vol. 21 No. 5. P. 673-678.
4. Barkalov K.A., Gergel V.P. Multilevel scheme of dimensionality reduction for parallel global search algorithms. // OPT-i: Proceedings of the 1st International Conference on Engineering and Applied Sciences Optimization 2014. P. 2111-2124.
5. Gergel V., Grishagin V., Israfilov R. Local tuning in nested scheme of global optimization. // Procedia Computer Science, 2015. Vol. 51, P. 865-874.
6. Gergel V., Grishagin V., Gergel A. Adaptive nested optimization scheme for multidimensional global search. Journal of Global Optimization, 2015. 17 P. Article in Press.
7. Barkalov K., Gergel V. Parallel global optimization on GPU // Journal of Global Optimization. 2016. 18 P. Article in Press.
8. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. М.: Физматлит, 2008. 352 с.
9. Gaviano M., Lera D., Kvasov D. E., Sergeyev Y. D. Software for generation of classes of test functions with known local and global minima for global optimization // ACM Transactions on Mathematical Software. 2003. Vol. 29. P. 469-480.
10. Strongin R.G., Sergeyev Ya.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht, Kluwer Academic Publishers, 2000.

11. Стронгин Р.Г., Гергель В.П., Гришагин В.А., Баркалов К.А. Параллельные вычисления в задачах глобальной оптимизации. Издательство М.: Московского университета. 2013. 280 с.
12. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация. Н.Новгород: Изд-во ННГУ, 2007.
13. Химмельблау Д. Прикладное нелинейное программирование. М.: Мир, 1975. 536 с.
14. Sergeyev Ya.D., Kvasov D.E. Global search based on efficient diagonal partitions and a set of Lipschitz constants, SIAM Journal on Optimization. 2006 No. 3. Vol. 16. P. 910–937.
15. Сысоев А.В. Баркалов К.А. Гергель В.П. Лебедев И.Г. Решение задач глобальной оптимизации на гетерогенных кластерных системах // Суперкомпьютерные дни в России: Труды международной конференции (28-29 сентября 2015 г., г. Москва).Москва: Изд-во МГУ. 2015. С. 411-419
16. Сысоев А.В. Баркалов К.А. Гергель В.П. Лебедев И.Г. MPI-реализация блочной многошаговой схемы параллельного решения задач глобальной оптимизации // Суперкомпьютерные дни в России: Труды международной конференции (28-29 сентября 2015 г., г. Москва).– Москва: Изд-во МГУ. 2015. С. 61-68.

Solving global optimization problems on GPU*

K.A. Barkalov, I.G. Lebedev

Lobachevsky State University of Nizhni Novgorod

In present study an approach for solving global optimization problems is developed. This approach is based on nested optimization scheme. A modification of this scheme based on use of different algorithms on different nested levels is proposed: complex serial algorithm (on CPU) is used on upper level; simple parallel algorithm (on GPU) is used on lower level. This scheme of computations is implemented in parallel solver ExaMin. Results of numerical experiments that demonstrate speedup of the algorithm are presented.

Keywords: global optimization, multiextremal functions, dimension reduction, parallel algorithms, GPU.

References

1. Gergel V.P. A method of using derivatives in the minimization of multiextremum functions // Computational Mathematics and Mathematical Physics. 1996. Vol. 36 No. 6. P. 729–742.
2. Gergel V.P., Sergeyev Ya.D. Sequential and parallel algorithms for global minimizing functions with Lipschitzian derivatives // Computers and Mathematics with Applications 1999. Vol. 37 No. 4-5. P. 163–179.
3. Gergel V.P., Strongin R.G. Parallel computing for globally optimal decision making on cluster systems // Future Generation Computer Systems 2005. Vol. 21 No. 5. P. 673–678.
4. Barkalov K.A., Gergel V.P. Multilevel scheme of dimensionality reduction for parallel global search algorithms. // OPT-i: Proceedings of the 1st International Conference on Engineering and Applied Sciences Optimization 2014. P. 2111–2124.
5. Gergel V., Grishagin V., Israfilov R. Local tuning in nested scheme of global optimization. // Procedia Computer Science, 2015. Vol. 51, P. 865–874.
6. Gergel V., Grishagin V., Gergel A. Adaptive nested optimization scheme for multidimensional global search. Journal of Global Optimization, 2015. 17 P. Article in Press.
7. Barkalov K., Gergel V. Parallel global optimization on GPU // Journal of Global Optimization. 2016. 18 P. Article in Press.
8. Sergeyev Y.D., Kvasov D.E. Diagonal'nyye metody global'noy optimizatsii. Fizmatlit, 2008. P. 352.
9. Gaviano M., Lera D., Kvasov D. E., Sergeyev Y. D. Software for generation of classes of test functions with known local and global minima for global optimization // ACM Transactions on Mathematical Software. 2003. Vol. 29. P. 469–480.
10. Strongin R.G., Sergeyev Ya.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht, Kluwer Academic Publishers, 2000.
11. Strongin R., Gergel V., Grishagin V., Barkalov K. Parallel'nyye vychisleniya v zadachakh global'noy optimizatsii. Publishing of the Moscow State University. 2013. 280 P.
12. Gorodetskiy S., Grishagin V. Nonlinear programming and multiextremal optimization. Nizhni Novgorod, Publishing of the Nizhni Novgorod State University, 2007.
13. Himmelblau, D.M.: Applied Nonlinear Programming. New York, McGraw-Hill, 1972. 498 P.

* This study was supported by the Russian Science Foundation, project No 15-11-30022.

14. Sergeyev Ya.D., Kvasov D.E. Global search based on efficient diagonal partitions and a set of Lipschitz constants, SIAM Journal on Optimization. 2006 No. 3. Vol. 16. P. 910–937.

15. Sysoev A.V. Barkalov K.A. Gergel V.P. Lebedev I.G. Solving the global optimization problems on heterogeneous cluster systems // Russian Supercomputing Days: Proceedings of the International Scientific Conference (Moscow, Russia, September 28–29, 2015.) Moscow, Publishing of the Moscow State University 2015. P. 411–419.

16. Sysoev A.V. Barkalov K.A. Gergel V.P. Lebedev I.G. MPI implementation of dimension reduction multilevel scheme for parallel solving the global optimization problems // Russian Supercomputing Days: Proceedings of the International Scientific Conference (Moscow, Russia, September 28–29, 2015.) Moscow, Publishing of the Moscow State University 2015. P.61–68.