Решение задач глобальной оптимизации на гетерогенных кластерных системах*

К.А. Баркалов, В.П. Гергель, И.Г. Лебедев, А.В. Сысоев

Нижегородский государственный университет им. Н.И. Лобачевского

Представлены результаты исследования параллельного информационно-статистического алгоритма глобальной оптимизации, разработанного в ННГУ им. Н.И. Лобачевского. Данный метод комбинируется с блочной схемой редукции размерности, что позволяет решать многомерные задачи путем их сведения к параллельному решению информационно-независимых подзадач меньшей размерности. Новым элементом, реализованным в рамках проводимого исследования, является использование нескольких графических ускорителей, задействованных на разных вычислительных узлах. Приводятся результаты решения тестовых задач на суперкомпьютере «Лобачевский» с использованием десятков тысяч GPU ядер.

1. Введение

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

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

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, а некоторым алгоритмом вычисления ее значений в точках области D; при этом *испытание* (вычисление одного значения) является вычислительно-трудоемкой операцией.

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 (так называемое «проклятие размерности»).

Снижение объема вычислений может быть достигнуто при построении неравномерного покрытия области поиска: сетка должна быть достаточно плотной в окрестности глобального оптимума и более редкой вдали от искомого решения. Построение такого рода покрытий обес-

* Работа поддержана грантом МОН РФ (соглашение от 27 августа 2013 г. № 02.B.49.21.0003 между МОН РФ и ННГУ ни. Н.И. Лобачевского).

печивается при повышении сложности самих численных методов глобального поиска. Применение неравномерных покрытий позволяет повысить размерность решаемых задач глобальной оптимизации в 2-3 раза, что является критичным в приложениях. Получаемые оценки размерности (20-30) позволяют охватить большинство научно-прикладных задач, поскольку, как правило, количество варьируемых параметров, по которым проявляется многоэкстремальность оптимизируемых критериев, ограничено.

Рассмотрим теперь основные способы распараллеливания вычислений, которые могут быть применены при решении задач глобальной оптимизации.

Во-первых, можно организовать разделение области решения между процессорами и параллельно решать подзадачи в этих подобластях. Однако такой подход обладает низкой эффективностью, поскольку при разделении области поиска только небольшая часть процессоров (в худшем случае – только один из них) будет решать задачу в подобласти с искомым глобальным минимумом; остальные процессоры будут работать в подобластях, в которых отсутствует решение исходной задачи.

Во-вторых, можно распараллеливать вычисление целевой функции, описывающей оптимизируемый объект. Данный путь может давать ускорение, но является специфичным для каждой конкретной решаемой задачи.

В-третьих, можно распараллелить реализацию вычислительных правил алгоритма, обеспечивающих выбор точки проведения очередного испытания. В этом случае способ распараллеливания будет зависеть от конкретного класса алгоритмов и, кроме того, часто эти правила достаточно просты и распараллеливать их нецелесообразно (накладные расходы на организацию параллелизма могут свести к нулю возможное ускорение).

Наконец, можно изменить схему алгоритма с целью параллельного выполнения нескольких испытаний (именно этот подход и будет рассматриваться в данной работе). Он является наиболее перспективным, т.к. характеризуется эффективностью (распараллеливается именно та часть вычислительного процесса, в котором выполняется основной объем вычислений) и общностью (применим для широкого класса характеристических алгоритмов многоэкстремальной оптимизации). Одновременно данный подход позволяет эффективно задействовать гетерогенные вычислительные ресурсы современных суперкомпьютеров (ядра на центральном процессоре, графические ускорители, математические сопроцессоры).

Данная статья продолжает серию исследований, начальные результаты которых были отражены в [1, 2].

2. Базовый параллельный алгоритм глобального поиска

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. Поэтому 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 $\varphi(x)$ 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

$$X_k = \{x^1,...,x^k\} \cup \{0\} \cup \{1\}$$

которое включает в себя граничные точки интервала [0,1], а также точки предшествующих испытаний, нижними индексами в порядке увеличения значений координаты, т.е.

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

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

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

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

Правило 3. Для каждого интервала $(x_{i-1}, x_i), 1 \le i \le k+1$, 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 $R(i), 1 \le i \le k+1$, упорядочить в порядке убывания

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

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

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

$$x^{k+j} = \frac{x_{t_j} + x_{t_j - 1}}{2}, \; t_j = 1, \; t_j = k+1$$

$$x^{k+j} = \frac{x_{t_j} + x_{t_j - 1}}{2} - \frac{z_{t_j} - z_{t_j - 1}}{2M}, \quad 1 < t_j < k+1. \tag{7}$$

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

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

Обоснование данного способа организации параллельных вычислений приведено в работах [3, 4]. Наиболее важным здесь является то, что 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 [4].

3. Редукция размерности

Одним из подходов к решению многомерных задач глобальной оптимизации является сведение их к одномерным и использование эффективных одномерных алгоритмов глобального поиска к редуцированной задаче. Данном разделе будут кратко изложены два известных способа редукции размерности, а также их обобщение.

3.1 Редукция размерности с использованием кривых Пеано

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 [3]. 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

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

3.2 Рекурсивная схема редукции размерности

Схема рекурсивной оптимизации основана на известном (см. [5]) 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

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

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).

3.3 Блочная рекурсивная схема редукции размерности

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

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

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

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

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 на гетерогенном кластере

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.

Для описания параллелизма схемы рекурсивной оптимизации введем вектор распараллеливания

$$\pi = (\pi_1, \pi_2, \ldots, \pi_M), \tag{15}$$

где $\pi_i, 1 \le i \le M,$ обозначает число параллельно решаемых подзадач (i+1)-го уровня вложенности, возникающих в результате выполнения параллельных итераций на i-м уровне. Для M-го уровня число π_M означает количество параллельных испытаний в процессе минимизации функции $\varphi_M(u_1,\ldots,u_M) = \varphi(y_1,\ldots,y_N)$ по переменной u_M при фиксированных значениях $u_1,\ldots,u_{M-1}$, т.е. количество параллельно вычисляемых значений целевой функции $\varphi(y)$.

В общем случае величины $\pi_i, 1 \le i \le M,$ могут зависеть от различных параметров и меняться в процессе оптимизации, но мы ограничимся случаем, когда все компоненты вектора (15) постоянны. Применение параллельных характеристических алгоритмов в соответствии с блочной рекурсивной схемой и вектором распараллеливания (15) позволяет использовать для решения задачи (1)

$$\Pi = 1 + \sum_{i=1}^{M} \prod_{j=1}^{i} \pi_j ,$$

параллельно работающих процессоров/ядер.

Используя различные параметры вектора распараллеливания можно адаптировать алгоритм для работы на гетерогенной вычислительной системе. При этом операцией, которую можно эффективно реализовать на ускорителе, является параллельное вычисление сразу многих значений целевой функции. Пересылки данных от CPU к GPU будут минимальные: требуется лишь передать на GPU координаты точек испытаний, и получить обратно значения функции в этих точках. Обработка результатов испытаний в соответствии с алгоритмом, требующая работы с большим объемом накопленной поисковой информацией, может быть эффективно реализована на CPU.

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

$$\varphi_i(u_1,\ldots,u_i) = \min_{u_{i+1} \in D_{i+1}} \varphi_{i+1}(u_1,\ldots,u_i,u_{i+1})$$

при i=1,…,M–2 решаются только с использованием CPU. Непосредственно в данных подзадачах вычислений значений оптимизируемой функции не происходит: вычисление значения функции $\varphi_i(u_1,\ldots,u_i)$ – 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

$$\varphi_{M-1}(u_1,\ldots,u_{M-1}) = \min_{u_M \in D_M} \varphi_M(u_1,\ldots,u_M)$$

отличается от всех предыдущих подзадач – в ней происходит вычисление значений оптимизируемой функции, т.к. $\varphi_M(u_1,\ldots,u_M) = \varphi(y_1,\ldots,y_N)$. Данная подзадача также решается на CPU, но вычисление значений функции производится на ускорителе. При этом на CPU выполняются правила 1 – 4 параллельного алгоритма глобального поиска. Координаты p точек испытаний, вычисленные на шаге 4 алгоритма, накапливаются в промежуточном буфере, а затем передаются на графический процессор. На GPU происходит вычисление значений функции в этих точках, после чего результаты испытаний (снова через промежуточный буфер) передаются на CPU.

Самый простой вариант использования данной схемы будет соответствовать двухкомпонентному вектору распараллеливания $\pi = (\pi_1, \pi_2)$. Здесь $\pi_1 + 1$ будет соответствовать числу

MPI-процессов, а π_2 – количеству используемый ядер на GPU; тем самым общее количество задействованных ядер (CPU и GPU) будет определяться как $1+\pi_1+\pi_1\pi_2$.

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

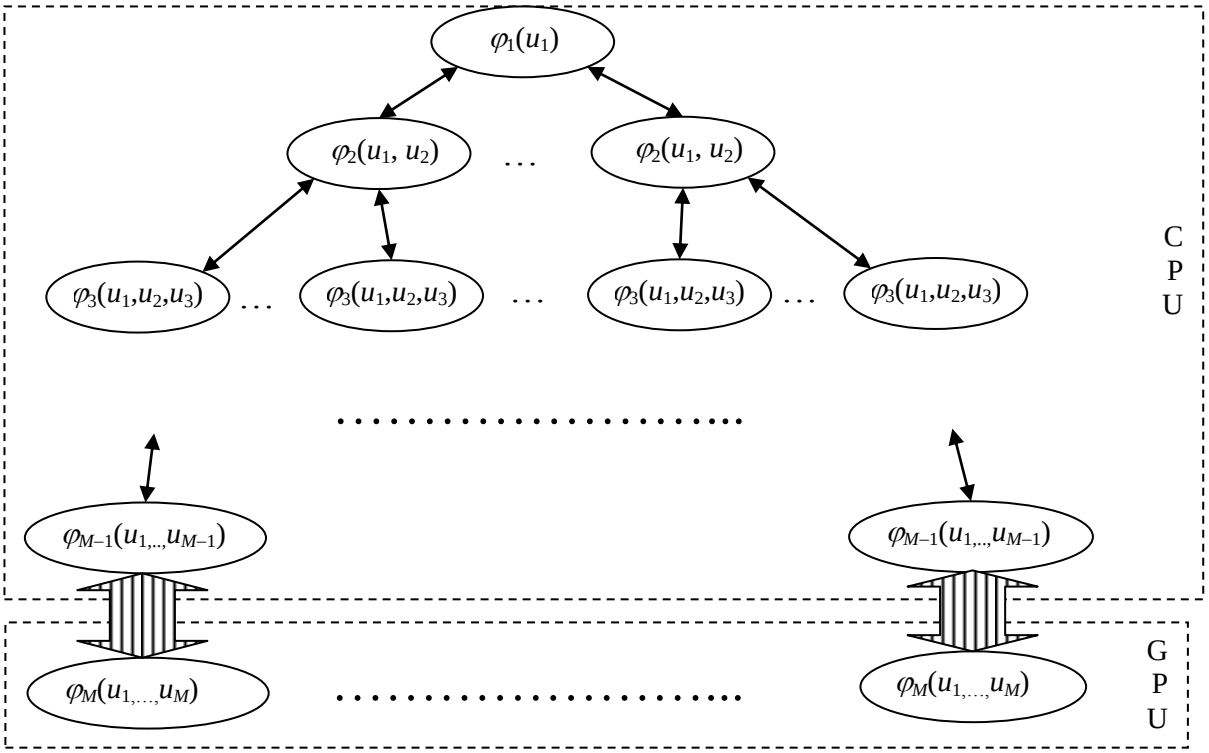

Рис. 1. Схема организации параллельных вычислений на кластере

5. Результаты вычислительных экспериментов

Вычислительные эксперименты проводились на суперкомпьютере «Лобачевский» (операционная система – CentOS 6.4, система управления – SLURM). Один узел суперкомпьютера располагает 2-я процессорами Intel Sandy Bridge E5-2660 2.2 GHz, 64 Gb RAM, и 2-я GPU NVIDIA Kepler K20X. Центральный процессор является 8-и ядерным (т.е. всего на узле доступно 16 ядер CPU), а на графическом процессоре доступны 14 потоковых мультипроцессоров (2688 CUDA-ядер). Использовался компилятор Intel C++ 14.0.2 и CUDA Toolkit 6.0.

В качестве тестовых задач были выбраны задачи, порождаемые с помощью GKLS-генератора [6]. Данный генератор позволяет получать задачи многоэкстремальной оптимизации с заранее известными свойствами: количеством локальных минимумов, размерами их областей притяжения, точкой глобального минимума, значением функции в ней и т.п. С целью имитации вычислительной трудоемкости, присущей прикладным задачам оптимизации, расчет целевой функции во всех проводимых экспериментах был усложнен дополнительными вычислениями, не меняющими вид функции и расположение ее минимумов (суммированием ряда из 20 тыс. элементов).

Для примера на рис. 2 приведены линии уровня двумерной функции, порождаемой GKLS-генератором. Темные точки соответствуют 464 испытаниям, потребовавшимся для решения данной задачи с точностью 0.01 по координате; при этом число итераций составило 58 (на каждой итерации проводилось 8 испытаний). Линии уровня наглядно демонстрируют многоэкстремальность задачи, а расположение точек – неравномерное покрытие, сгущающеюся только в районе глобального минимума.

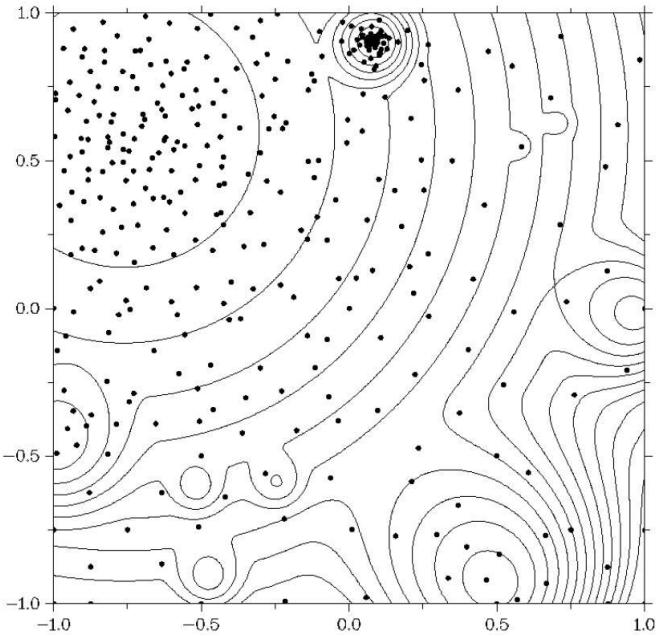

Рис. 2. Решение двумерной задачи

Эксперимент с использованием одного графического ускорителя проведем для решения серии из 100 шестимерных задач. Глобальный минимум y^* в тестовой задаче будем считать найденным, если алгоритм сгенерировал точку испытания y^k в δ-окрестности глобального минимума, т.е. $\left\|y^k - y^*\right\| \le \delta$. Размер окрестности $\delta = 0.01\|b-a\|$, где a и b – границы области поиска D. Параметр надежности метода был выбран $r = 4.5$; параметр построения развертки – $m = 10$. Максимально допустимое число параллельных итераций составляло K_{max} = 10 000. Так как задача решалась с использованием единственного ускорителя, то разбиения размерностей не проводилось.

Среднее время решения задачи с использованием GPU составило 10.78 с., тогда как среднее время решения задачи с использованием всех 16 ядер CPU на узле – 53.8 с.; наблюдается почти пятикратное ускорение.

Более масштабный вычислительный эксперимент был проведен для задач размерности N = 8 и N = 10: было использовано 12 узлов кластера с 36 графическими ускорителями (по три ускорителя на узел), тем самым было задействовано 96 768 CUDA-ядер.

В соответствии с блочной рекурсивной схемой (13) было использовано два уровня подзадач с размерностями $N_1 = N_2 = 4$ для восьмимерной задачи и $N_1 = N_2 = 5$ для десятимерной. Вектор распараллеливания (15) был выбран как $\pi = (36,2688)$ в соответствии с общим числом используемых GPU (компонента π_1) и CUDA-ядер на одном ускорителе (компонента π_2). Время решения восьмимерной задачи составило 405.6 с., ускорение по сравнению с CPU версией алгоритма составляет 5.9 раз. Время решения десятимерной задачи составило 2055.8 с. Десятимерная задача ввиду большой сложности вычислений на CPU не решалась, поэтому оценить ускорение не представляется возможным.

6. Заключение

Результаты проведенных экспериментов на серии тестовых задач разной размерности показывают, что предложенная блочная многошаговая схема редукции размерности в сочетании с параллельным алгоритмом глобального поиска эффективно реализуется на современных вычислительных системах. Данная схема обладает:

- высоким запасом параллелизма (было задействовано порядка 10^5 ядер);
- малой информационной зависимостью параллельно выполняемых вычислений (между параллельными процессами нужно было передавать лишь точки и результаты испытаний, по-

лученные на текущей итерации метода).

Полученные результаты позволяют сделать предположение, что предложенная схема организации параллельных вычислений будет эффективна и при использовании большего (порядка 10^6) числа ядер. Дальнейшее развитие также будет заключаться в использовании локальной информации о поведении оптимизируемой функции [7], учете ограничений в задачах условной оптимизации [8], адаптации алгоритмов для решения многокритериальных задач [9].

Литература

1. K. Barkalov, V. Gergel. Multilevel scheme of dimensionality reduction for parallel global search algorithms // OPT-i 2014. An International Conference on Engineering and Applied Sciences Optimization (Kos Island, Greece, 4–6 June 2014). 2014. pp. 2111–2124.
2. И.Г. Лебедев, К.А. Баркалов. Реализация параллельного алгоритма глобального поиска на GPU // Вестник ПНИПУ. Аэрокосмическая техника. 2014. № 39. С. 64–82.
3. R.G. Strongin, Ya.D. Sergeyev, Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht, 2000.
4. Стронгин Р.Г., Гергель В.П., Гришагин В.А., Баркалов К.А. Параллельные вычисления в задачах глобальной оптимизации. М.: Издательство Московского университета. 2013. 280 с.
5. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация. Н.Новгород: Изд-во ННГУ, 2007.
6. Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. М.: Физматлит, 2008.
7. Баркалов К.А., Рябов В.В., Сидоров С.В. О некоторых способах балансировки локального и глобального поиска в параллельных алгоритмах глобальной оптимизации // Вычислительные методы и программирование: новые вычислительные технологии. 2010. Т. 11. № 1. С. 382–387.
8. Маркин Д.Л., Стронгин Р.Г. О равномерной оценке множества слабоэффективных точек в многоэкстремальных многокритериальных задачах оптимизации // Ж. вычисл. матем. и матем. физ., 1993. Т. 33, №2. С. 195–205
9. Баркалов К.А., Стронгин Р.Г. Метод глобальной оптимизации с адаптивным порядком проверки ограничений // Ж. вычисл. матем. и матем. физ. 2002. Т.42, №9. С. 1338–1350.

Solving the global optimization problems on heterogeneous cluster systems

Konstantin Barkalov, Victor Gergel, Ilya Lebedev and Alexander Sysoyev

Keywords: global optimization, parallel algorithms, heterogeneous computing

The paper presents the results of investigation of parallel information-statistical algorithm of global optimization that developed in University of Nizhny Novgorod. This method is combined with multilevel scheme of dimension reduction. It allows us to solve multidimensional problems by reducing it to parallel solving a series of data-independent subtasks of lower dimension. New suggestion developed in the framework of current research is the method of using several graphics accelerators on different nodes of cluster. Results of solving a series of test problems on supercomputer Lobachevsky using tens of thousands GPU cores are given.