Исследование эффективности глобальной параллельной оптимизации функций многих переменных

А.Н. Коварцев, Д.А. Попова-Коварцева, П.В. Аболмасов

Самарский государственный аэрокосмический университет
им. академика С.П. Королёва (национальный исследовательский университет)

Рассматривается параллельный алгоритм глобальной оптимизации функций многих переменных модифицированным методом половинных делений, основанный на использовании локальной техники. Основное внимание уделяется рациональной организации вычислений за счет распределения вычислительных ресурсов между фазами глобальной и локальной оптимизации. При разработке алгоритмов использовалось визуальное средство автоматизации программирования параллельных вычислений PGRAPH. Представлены результаты численного тестирования предложенного алгоритма.

1. Введение

Проблема разработки эффективных методов решения задачи глобальной оптимизации функций многих переменных является чрезвычайно важной для науки и техники. В этой области накоплен значительный опыт и разработано большое количество алгоритмов и методов решения задач многоэкстремальной оптимизации. В то же время, преодоление принципиальной трудности задач глобальной оптимизации, связанной с экспоненциальным ростом их сложности в зависимости от размерности оптимизируемой функции, остаётся открытой проблемой, во всяком случае, для точных методов оптимизации.

Современные точные методы глобальной оптимизации совершенствуются, главным образом, для класса Липшицевых функций в двух направлениях: первое связано с разработкой новых критериев отсева неперспективных областей оптимизируемой функции [1], второе - с повышением эффективности методов редукции задачи многомерной оптимизации [2]. В работе [3] на примере метода неравномерных покрытий и его модификаций была предпринята попытка качественного анализа эффективности методов решения задачи глобальной оптимизации. В частности получена следующая оценка средней сложности задачи глобальной оптимизации: $\widetilde{N}(n) = \beta \cdot \left(\frac{1}{4\varepsilon} \right)^{n-\mu} - 1$, где $\beta = \frac{(\alpha+1)2^n - 2}{\alpha 2^n - 1}$, α $(\alpha > 1/2^n)$ – доля неотбракованных параллелепипедов ($(1-\alpha)$ параллелепипедов отбраковываются с помощью критериев отсева), $\mu = \lfloor \log_2 \alpha \rfloor$, ε - заданная точность решения задачи глобальной оптимизации. При $\alpha = 1$, когда критерии отсева параллелепипедов не работают, алгоритмы глобальной оптимизации имеют самую низкую производительность и практически «не работают», начиная с $n > 5$. Для предельно возможной эффективности правил отсева ($\alpha \approx 1/2^{n-1}$, $\mu \approx n-1$) практическое применение прямых методов глобальной оптимизации ограничено ($n < 20$). Не лучше складывается ситуация для методов редукции задачи оптимизации. В [3] показано, что в этом случае справедлива следующая оценка средней сложности задачи глобальной оптимизации: $\widetilde{N}(n) = \left(\beta \cdot \left(\frac{1}{4\varepsilon} \right)^{1-\mu} - 1 \right)^n$, т.е. сохраняется экспоненциальный характер роста этого показателя. В то же время, для реальных значений α ($\alpha \approx 0.4...0.8$) средняя сложность алгоритмов глобальной оптимизации для метода редукции на несколько порядков меньше, чем для методов, не использующих редукцию.

В последнее время решение проблемы высокой вычислительной сложности задач глобальной оптимизации связывают с использованием методов параллельных и распределенных вычислений [6]. Однако, использование этой относительно «новой» техники вычислений на прак-

тике позволяет получить как существенное ускорение алгоритма глобальной оптимизации, так и замедление его характеристик. Как для любого «нового дела», процесс распараллеливания алгоритмов глобальной оптимизации имеет свои «подводные камни», особенности и специфические трудности. В данной работе на примере модифицированного метода половинных делений [4] рассматриваются некоторые особенности организации параллельных алгоритмов глобальной оптимизации.

2. Постановка задачи и основные определения

Рассмотрим задачу безусловной глобальной оптимизации непрерывной функции $f : R^n \to R$, заданной на допустимом множестве $X \in R^n$ в следующей постановке:

$$f_* = glob\min_{x \in X} f(x) = f(x_*) . \quad (1)$$

Положим, что глобальный минимум x_* принадлежит множеству X_*, причем $X_* \subset X$, а $X = \overset{n}{\underset{i=1}{\otimes}} [0,1]$, является многомерным единичным гиперкубом. Предлагаемый далее алгоритм глобальной оптимизации является модификацией известного метода половинных делений (метода неравномерных покрытий) [1], основные положения которого заключаются в следующем.

Определим множество ε - решений задачи (1) как

$$X_*^{\varepsilon} = \{ x \in X : f(x) \le f_* + \varepsilon \} . \quad (2)$$

Нахождение приближенного решения задачи (1) заключается в поиске хотя бы одной точки из множества X_*^{ε}.

В методе неравномерных покрытий строится последовательность $\{X_i\} = \{X_1,...,X_k\}$ подмножеств множества X и точек $x_1,...,x_k$ ($x_i \in X_i$) (для метода половинных делений в качестве X_i обычно рассматриваются параллелепипеды). В каждой точке x_i вычисляется значение функции $f(x)$ и определяется её рекордное значение f_r:

$$f_r = \min_{1 \le i \le k} f(x_i) = f(x_r), \quad 1 \le r \le k . \quad (3)$$

На каждом из подмножеств X_i вводятся *миноранты*, т.е. такие функции $G_i(x)$, что $\forall x \in X_i \quad f(x) \ge G_i(x)$. Тогда последовательности подмножеств $\{X_i\}$ можно поставить в соответствие совокупность покрывающих подмножеств $\{Z_i\} = Z_1,..., Z_k$, определяемых из условий

$$Z_i = \{ x \in X_i : G_i(x) \ge f_r - \varepsilon \}, \; r \le i . \quad (4)$$

Множество Z_i строится таким образом, что $\forall x \in Z_i$ выполняется неравенство $f(x) \ge f_r - \varepsilon$, из чего следует, что глобальный минимум функции $f(x)$ на множестве Z_i не может быть меньше рекорда f_r более чем на ε. Таким образом, в процессе поиска глобального минимума подмножество Z_i можно исключить из рассмотрения и продолжить оптимизацию на множестве $X_i \setminus Z_i$. Алгоритм останавливается, когда допустимое множество оказывается покрытым подмножествами Z_i, т.е.

$$X \subseteq Z_1 \cup Z_2 \cup ... \cup Z_k . \quad (5)$$

Алгоритм неравномерных покрытий гарантирует нахождение ε-оптимального решения, что подтверждается соответствующей теоремой.

Предлагаемая модификация метода половинных делений связана с введением понятия *области притяжения локального минимума* и использованием локальной техники. Пусть $x_i^*, i = 1,.., m$ – локальные минимумы функции $f(x)$. Тогда область притяжения локального минимума x_i^* определим как множество точек начальных приближений алгоритма локальной оптимизации, из которых алгоритм сходится к x_i^*:

$$S_i = \{ x \in X : \ arg\ min f(x) = x_i^* \}. \quad (6)$$

Поскольку построение области притяжений в смысле (6) достаточно сложная задача, далее будем использовать следующее упрощенное определение области притяжения локального минимума:

$$S_i = \{ x \in X : \ || x - x_i^* || < \rho_i, \quad \rho_i > 0 \}, \quad (7)$$

где ρ_i – радиус области притяжения локального минимума.

2.1 Использование локальной техники

Для модифицированного алгоритма половинных делений сохраняется экспоненциальный рост сложности задачи оптимизации в зависимости от числа переменных. Одной из эффективных стратегий совершенствования алгоритмов глобальной оптимизации (ГО) является использование *локальной техники*, когда стратегия глобального поиска удачно сочетается с методами локальной оптимизации (ЛО).

При ограниченном количестве минимумов функции $f(x)$ области притяжения S_i имеют значительные размеры. Предположим, что относительно оптимизируемой функции известно количество локальных минимумов – r, включая глобальный, и размеры областей притяжения: $\rho_1,...,\rho_r$. Введем понятие «гарантированного радиуса» области притяжения минимумов функции, под которым будем понимать $\rho_m = min\ \rho_i, \quad 1 < i \le r$.

В этом случае, как показано в работе [3], среднюю сложность алгоритма глобальной оптимизации можно оценить величиной

$$\widetilde{N}(n) = 2\left(1/4\rho_m\right)^n - 1 + r\, log_2(\rho_m / \varepsilon). \quad (8)$$

С учетом (8) при $\rho_m >> \varepsilon$ этап глобальной оптимизации, связанный с поиском областей притяжения локальных минимумов функции, можно производить достаточно грубо, а следовательно, с меньшими затратами времени.

2.2 Двухфазный алгоритм метода половинных делений (ДАМПД)

Дополнительная информация о размерах областей притяжения локальных минимумов функции позволяет построить двухфазную схему алгоритма глобальной оптимизации, в которой выделим фазы глобальной и локальной оптимизации (см. рис. 1).

В ДАМПД фаза глобальной оптимизации реализуется с помощью модифицированного алгоритма, с той лишь разницей, что двоичное деление параллелепипедов реализуется до достижения заданных размеров их радиусов R_i. Величина R_i определяется размером «гарантированного радиуса» ρ_m областей притяжения минимумов функции. На рисунке 1 пунктирными линиями отмечены гарантированные области притяжения минимумов функции. Заштрихованные прямоугольники являются прямоугольниками, «отбракованными» с использованием константы Липшица. В итоге, исходя из условия $R_i \le \rho_m$, фаза глобальной оптимизации завершилась, породив, для приведенного на рисунке 2 примера, 24 прямоугольника заданного радиуса.

В фазе локальной оптимизации из точек, принадлежащих областям притяжения локальных минимумов, организуется поиск минимумов функции с помощью локальных алгоритмов оптимизации.

Такая схема организации процедуры поиска глобального минимума функции существенно сокращает трудоемкость фазы глобальной оптимизации.

В работе [4] исследована сходимость двухфазного алгоритма метода половинных делений, где показано, что для дважды дифференцируемой Липшицевой функции, вторая производная которой также удовлетворяет условию Липшица, ДАМПД обеспечивает нахождение ε-оптимального решения.

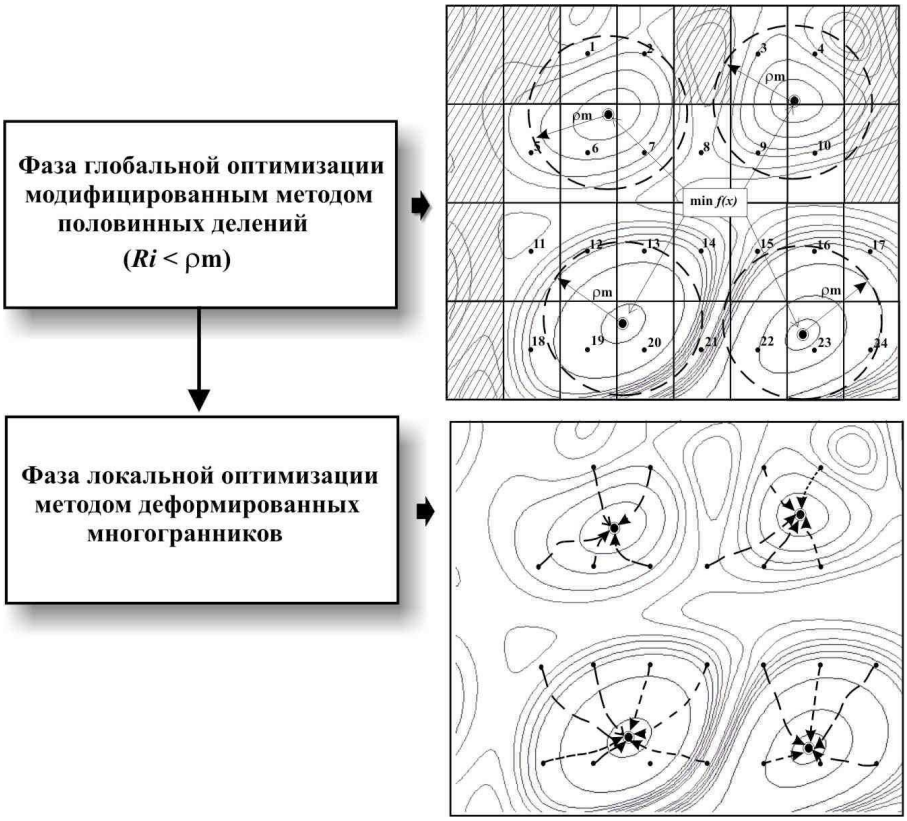

Рис. 1. Двухфазный алгоритм метода половинных делений

3. Алгоритм формирования множества точек начальных приближений для ДАМПД

Для больших размерностей вектора независимых переменных оптимизируемой функции в фазе глобальной оптимизации порождается значительное количество параллелепипедов заданного размера. В этом случае объем вычислений в фазе локальной оптимизации становится чрезмерно большим. Рассмотрим следующий адаптивный алгоритм формирования списка точек начальных приближений областей притяжения локальных минимумов оптимизируемой функции, осуществляющий сжатие количества точек начальных приближений, используемых во второй фазе алгоритма.

Будем считать, что область притяжения S_i определена, если она содержит, по крайней мере, одну точку из множества $C = \{c_1,...,c_k\}$, где c_k - центры параллелепипедов.

Пусть r – количество зон притяжения минимума функции; ρ_m – гарантированный радиус области притяжения минимума функции, обеспечивающий нахождение всех локальных минимумов.

Сформируем список областей притяжения локальных минимумов $V = \{V_1,V_2,...,V_m\}$, элементами которого являются структуры $V_i = (\widetilde{c}_i, \widetilde{f}_i)$, где $\widetilde{c}_i$ - координаты «представителя» i-й области притяжения, имеющего наилучшую, достигнутую для этой области, оценку $\widetilde{f}_i$. Вектор $\widetilde{c}_i$ условно считается центром i-ой области притяжения.

Первоначально список V пуст. По мере вычислений функции он наполняется элементами, но в конце этапа глобального поиска не может содержать больше m элементов (m – заданный размер списка, $m \geq r$). Размер списка m зависит от свойств оптимизируемой функции и выбирается из соображений попадания в него представителя области притяжения глобального минимума функции. Элементы списка V упорядочены таким образом, что $\widetilde{f}_1 < \widetilde{f}_1 < ... < \widetilde{f}_m$.

Пусть в процессе работы алгоритма глобальной оптимизации произведено очередное испытание $f_k = f(c_k)$. Эволюция содержимого списка V происходит по следующим простым правилам:

1. Проверяется принадлежность центра очередного параллелепипеда c_k окрестностям одной из имеющихся областей притяжения $V_i \quad i = 1,...,l$ (l – текущий размер списка).

1.1.Если выполняется условие

$$c_k \in \{ x \in X : ||(x - \tilde{c}_i)|| \le \rho_m \}. \qquad (9)$$

1.1.1. То при $f_k < \tilde{f}_i$ содержимое элемента V_i обновляется:

$\tilde{c}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} c_j$, $\tilde{f}_i := f_k$, где c_j - центры параллелепипедов «попавших» в область V_i.

Далее выполняется действие 1.1.3.

1.1.2. Иначе уточняется только значение $\tilde{c}_i$. Выполняется действие 1.1.3.

1.1.3. Список V упорядочивается в порядке возрастания $\tilde{f}_i$.

1.2. Если условие (9) не выполняется, то проверяем правило 2.

2. Определяется возможность включения нового элемента в список V.

2.1. Если $(f_k < \tilde{f}_1)$, то элемент V_k записывается в голову списка V.

2.2. Если $(\tilde{f}_j < f_k \le \tilde{f}_{j+1})$, то элемент V_k записывается между V_j и V_{j+1} элементами списка.

2.3. Если $(f_k > \tilde{f}_l)$, то элемент V_k записывается в конец списка V.

3. При превышении предельного числа элементов списка, из списка исключается последний элемент списка.

Предложенный алгоритм является эвристическим, однако, вычислительные эксперименты с тестовыми функциями показали его состоятельность. Более того, как правило, в первой сотне элементов списка содержится начальное приближение глобального минимума функции. На рисунке 2 представлены элементы списка областей притяжения. Нумерация соответствует весу элемента. Из рисунка видно, что каждый элемент списка сформировался, агрегируя большое количество точек вычисления функции. Например, элемент V_1 агрегировал 9 точек, V_2 - 8, и т.д.

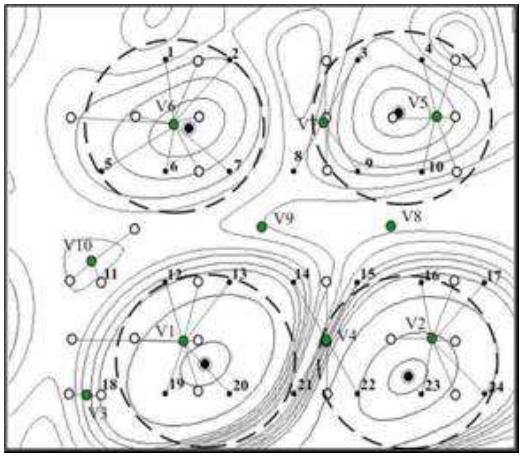

Рис. 2. Формирование списка областей притяжения

4. Визуальное средство реализации параллельных алгоритмов PGRAPH

Методы разработки параллельных алгоритмов существенно отличаются от методов последовательного программирования. Здесь большое значение приобретают средства автоматиза-

ции разработки моделей и кодов параллельных алгоритмов. Технология программирования (особенно в параллельном исполнении) должна быть понятна конечному пользователю, специалисту в предметной области.

4.1 Средство визуального программирования PGRAPH

При разработке и реализации двухфазного алгоритма метода половинных делений использовалась средство моделирования параллельных вычислений PGRAPH [5]. Система PGRAPH предоставляет пользователю возможность визуального формирования графических образов разрабатываемых алгоритмов, средства автоматизированного анализа проектируемых программ, а также средства автоматического синтеза исходных кодов программ. С помощью внешнего компилятора система позволяет компилировать и запускать на выполнение созданные в ней параллельные модели алгоритмов. При этом пользователь фактически «не видит» и не использует директивы MPI, которые автоматически «встраиваются» в текст программы при её компиляции.

Модель описания параллельных алгоритмов в системе PGRAPH представляется четверкой $< \Lambda, F, P, G >$ где Λ – множество данных некоторой предметной области, F – множество операторов, определенных над данными предметной области, P – множество предикатов, действующих над структурами данных предметной области, $G = (A, \Psi)$ – ориентированный помеченный граф, называемый агрегатом. $A = \{A_1, ..., A_n\}$ – множество вершин графа. Каждая вершина A_i помечена локальным оператором $\varphi_i(\Lambda) \in F$. На графе задано множество дуг управления $\Psi = \{\Psi_1, ..., \Psi_m\}$.

Дуга управления, соединяющая любые две вершины A_i и A_j, имеет три метки: *приоритет*; *предикат* – $p_{ij}(\Lambda) \in P$, управляющий ходом вычислительного процесса и *тип дуги*.

Граф G – инициальный. Работа алгоритма всегда начинается из начальной вершины. Дальнейшее развитие вычислений ассоциируется с переходами на графе из вершины в вершину по дугам управления. Для «последовательных» вершин, переход по дуге управления возможен лишь в случае истинности предиката, которым она помечена. Рассматриваются четыре типа дуг: *последовательная дуга* (описывает передачу управления на последовательных участках алгоритма) и обозначается на схеме прямоугольником; *параллельная дуга* (обозначает начало нового параллельного фрагмента алгоритма и помечается «кружочком»); *терминирующая дуга* (завершает параллельный фрагмент алгоритма, на схеме перечеркивается) и дуга синхронизации. Пример описания параллельного алгоритма представлен на рисунке 3.

4.2 Параллельный алгоритм глобальной оптимизации

Реализация алгоритма ДАМПД представлена на рисунке 3. Для удобства описания модели все вершины пронумерованы.

В вершине 1 устанавливаются значения параметров теста GKLS и производится его инициализация [7]. Алгоритм состоит из двух фаз. Первая фаза глобальной оптимизации состоит из вершин 2-5. В вершине 2 формируется начальный список параллелепипедов, которые впоследствии «раздаются» по процессорам кластера (см. вершину 3).

Глобальная оптимизация методом половинных делений реализуется в вершинах 4а-4г. На рисунке 3 для наглядности приведён вариант алгоритма для четырех процессоров, хотя несложно его масштабировать на любое число процессоров. Параллелепипеды делятся до тех пор, пока их радиус не станет меньше ρ_m. При этом на каждом процессоре формируются списки точек, лежащих в областях притяжения локальных минимумов. В вершине 5 сформировавшиеся списки объединяются.

Во второй фазе алгоритма осуществляет поиск локального минимума из точек, вычисленных на первом этапе. Данная фаза состоит из вершин 6-8. В вершине 6 происходит «раздача» точек начальных приближений процессорам для организации поиска локальных минимумов, который реализуется в вершинах 7а-7г. Вершина 8 завершает фазу локальной оптимизации. Алгоритм ДАМПД завершает свою работу по исчерпанию списка точек начальных приближений.

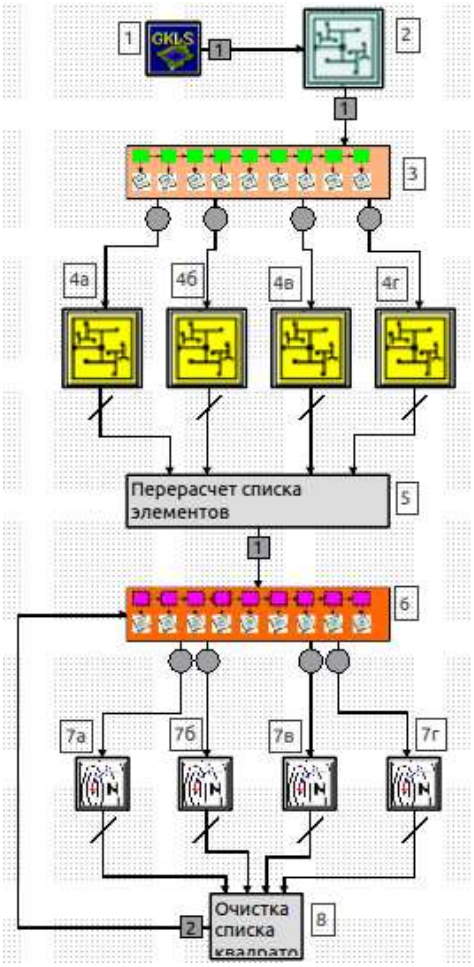

Рис. 3. Параллельная версия алгоритма ДАМПД

5. Вычислительные эксперименты

5.1 Условия тестирования

Для вычислительных экспериментов был выбран класс липшицевых функций, моделируемых известным генератором GKLS [7]. Генератор тестовых заданий GKLS порождает три класса тестовых функций: недифференцируемых, непрерывно дифференцируемых и дважды непрерывно дифференцируемых. Для каждого класса можно использовать 100 тестовых функций. В GKLS очень просто задать сложность генерируемых функций, определить количество локальных минимумов, размеры областей притяжения и многое другое.

Тестирование алгоритма ДАМПД проводилось на наиболее сложном для глобальной оптимизации, классе недифференцируемых функций. Для всех классов задач: число экстремумов равно десяти; радиус притяжения глобального оптимума – 0,33. Эксперименты проводились на суперкомпьютерном кластере СГАУ «Сергей Королев». Кластер построен на базе линейки оборудования IBM BladeCenter с использованием блейд-серверов HS22 и обеспечивает пиковую производительность более десяти триллионов операций с плавающей точкой в секунду. Общее

число процессоров/вычислительных ядер: 272/1184. Глобальный минимум вычислялся с точность $\varepsilon = 1{,}0 \cdot 10^{-8}$ (по аргументам функции), при этом использовалось 256 процессоров кластера.

5.2 Тестирование алгоритма ДАМПД

Результаты вычислительного эксперимента для алгоритма ДАМПД (при n=8) приведены в таблице 1. Общее ускорение алгоритма составило 91, хотя для фазы ЛО ускорение равно 139 . На рисунке 4 и 5 показана загрузка процессоров (количество обращений к оптимизируемой функции на каждом из них).

Таблица 1. Результаты эксперимента с базовой версией алгоритма ГО ДАМПД

Время работы алгоритма, сек	463,7
Число обращений к функции на этапе ГО (суммарно на всех процессорах)	262122 (23643168)
Число обращений к функции на этапе ЛО (суммарно на всех процессорах)	4922 (683665)

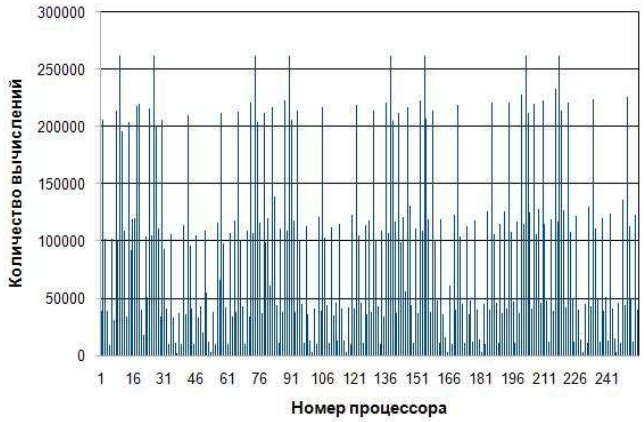

Рис. 4. Загрузка процессоров на этапе ГО в базовой версии

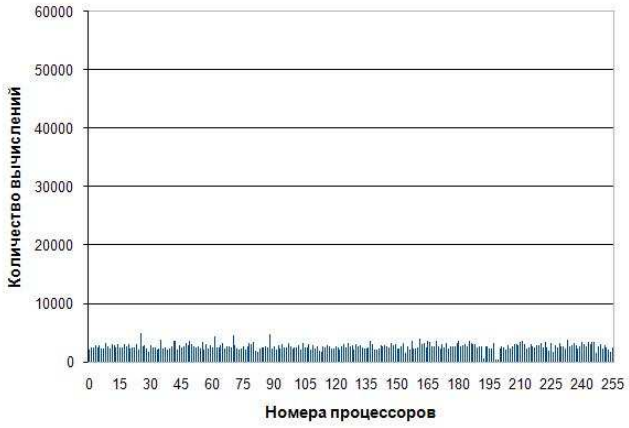

Рис. 5. Загрузка процессоров на этапе ЛО в базовой версии

Как видно из рисунков, наблюдается невысокая эффективность алгоритма ДАМПД для этапа глобальной оптимизации, поскольку процессоры кластера имеют неравномерную вычислительную нагрузку. Для фазы глобальной оптимизации количество делений параллелепипедов, реализуемое на каждом из процессоров, зависит от сложности доставшегося ему участка оптимизируемой функции. В итоге наблюдается нерациональное использование ресурсов кластера (используется только 35% потенциальной вычислительной мощности).

5.3 Тестирование алгоритма ДАМПД, реализованного по схеме «менеджер - исполнитель»

Для повышения эффективности алгоритма необходимо организовать адаптивное перераспределение нагрузки между процессорами, подкачивая по мере необходимости новые задания для «быстрых» процессоров.

Для этого в системе PGRAPH используются механизмы синхронизации параллельных процессов. Один из процессоров выделяется как управляющий (менеджер), а в его функции входит «раздача» заданий на оптимизацию для процессоров-исполнителей. Пример такой вычислительной схемы для этапа ЛО представлен на рисунке 6.

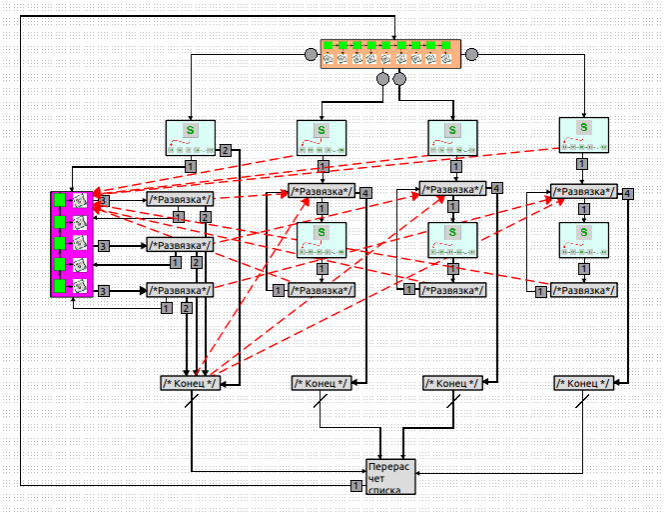

Рис. 6. Этап ЛО в режиме «менеджер – исполнитель»

Положим, что в фазе ЛО подготовлен список точек начальных приближений. Первоначально все процессоры, включая «менеджера», одновременно запускаются на поиск локального минимума. После завершения решения оптимизационной задачи ветвь «менеджера» переходит в режим диспетчера и ждет сообщения о завершении ЛО от остальных ветвей. На схеме передача сообщения изображается пунктирной стрелкой. После приёма сообщения от любой из параллельной ветви ветвь-менеджер передаёт соответствующему процессору новую точку начального приближения (фактически новое задание), запуская при этом очередной поиск локального минимума. Как только список точек начальных приближений заканчивается, ветвь-менеджер выходит из цикла приёма сообщений, и все ветви завершают свою работу.

Для фазы ГО асинхронное управление процессорами реализуется по аналогичной схеме. Необходимо лишь на этапе формирования начального списка параллелепипедов подготовить параллелепипедов больше, чем число процессоров, обеспечивая запас заданий для «быстрых» процессоров.

Результаты вычислительного эксперимента для данной версии алгоритма приведены в таблице 2.

Таблица 2. Результаты эксперимента для схемы «менеджер-исполнитель»

Время работы алгоритма, сек	268,2
Число обращений к функции на этапе ГО (суммарно на всех процессорах)	132662 (23715292)
Число обращений к функции на этапе ЛО (суммарно на всех процессорах)	12565 (2519983)

Увеличение ускорения для второго варианта ДАМПД объясняется тем, что для схемы «менеджер - исполнитель» в фазе глобальной оптимизации можно в разы увеличить начальный список параллелепипедов. При этом, естественно, уменьшаются их размеры (диаметры параллелепипедов) и, как следствие, уменьшается трудоемкость глобальной оптимизации на каждом из процессоров кластера, поскольку оптимизация производится до достижения заданного размера параллелепипеда сравнимого с «гарантированным радиусом» области притяжения. В этом случае получается, что максимальное время ГО на одном из процессоров во второй схеме

ДАМПД становится меньше, чем в первой схеме ($T_p^{(2)} < T_p^{(1)}$). Пусть трудоемкость последовательного алгоритма ГО составляет T_1. Если предположить, что обработка «избыточных» начальных параллелепипедов во второй схеме перераспределяется «менеджером» между освобождающимися процессорами, и суммарная загрузка процессоров не превышает $T_p^{(2)}$, то очевидно, что ускорение для второй схемы ДАМПД $S_p^{(2)} = T_1 / T_p^{(1)}$ будет больше, чем у первой схемы ($S_p^{(2)} > S_p^{(1)}$).

Общее ускорение составило 181. Загрузка процессоров для фаз ГО и ЛО показана на рисунке 7. В данном случае наблюдается более равномерное распределение нагрузки между процессорами. В частности общая эффективность алгоритма возросла до 70%, а фазы ЛО до 78%.

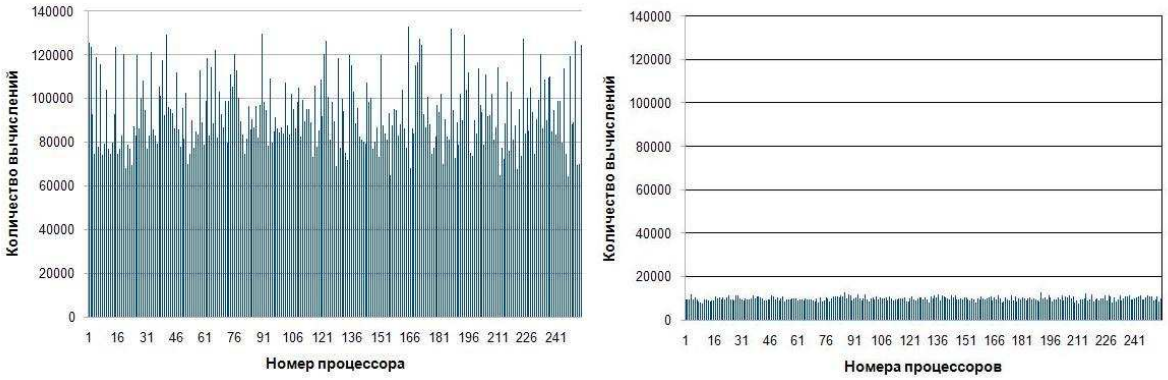

Рис. 7. Загрузка процессоров для этапов ГО и ЛО схемы «менеджер – исполнитель»

5.4 Зависимость сложности алгоритма ДАМПД от «гарантированного радиуса» притяжения

Несомненно, что трудоёмкость задачи глобальной оптимизации существенно зависит от величины «гарантированного радиуса» области притяжения. Последнее следует из формулы (8), поскольку с уменьшением радиуса ρ_m значительно увеличивается сложность фазы ГО, и немного уменьшается сложность фазы ЛО. На самом деле, сложность задачи оптимизации во многом определяется свойствами оптимизируемой функции. Например, «легче» отыскивается глобальный оптимум для функций, у которых области притяжения локальных экстремумов не пересекаются. Как показали вычислительные эксперименты, задача оптимизации становится более трудоёмкой, если области локальных притяжений её экстремумов вложены друг в друга.

Влияние радиуса притяжения на производительность алгоритма ДАМПД рассматривалась на примере задачи оптимизации функции 7 переменных для тестовой функции №4 GKLS (со стандартным набором установочных параметров) и для «хорошей» тестовой функции, описанной в работе [3]. В первом случае наблюдается вложенность областей локальных притяжений друг в друга. Во втором – области локальных притяжений экстремумов функции не пересекаются. Для того чтобы исключить влияние схемы реализации параллельных вычислений на оценку сложности задачи глобальной оптимизации, рассматривался последовательный алгоритм ДАМПД, а в качестве критерия оценки сложности использовалось число обращений к функции (N_1).

В таблицах 3 и 4 приведены результаты расчетов сложности последовательного алгоритма ДАМПД в зависимости от «гарантированного радиуса» области притяжения для тестовой функции GKLS и «хорошей» функции. Как видно из таблиц, в фазе ГО сложность задачи оптимизации стремительно увеличивается при уменьшении радиуса ρ_m. Однако значительно большее влияние на производительность алгоритма ДАМПД оказывают «топологические» свойства областей притяжения локальных экстремумов. Что же касается фазы ЛО, то сложность локальной оптимизации практически не зависит от радиуса притяжения.

Таблица 3. Сложность последовательного алгоритма ДАМПД для функции GKLS

Радиус притяжения, ρm	Число обращений к функции N1 на этапе	
	ГО	ЛО
0.31	1130068	115830
0.22	10361462	149928
0.16	60819698	291252

Таблица 4. Сложность последовательного алгоритма ДАМПД для «хорошей» функции [3]

Радиус притяжения, ρm	Число обращений к функции N1 на этапе	
	ГО	ЛО
0.31	45472	413199
0.22	725310	316044
0.16	4376072	283662

6. Заключение

В работе на примере задачи глобальной оптимизации продемонстрированы возможности средства визуального моделирования параллельных алгоритмов PGRAPH, ориентированного на конечного пользователя и освобождающего его от изучения непрофильных областей знаний, например, «тяжеловесного» стандарта MPI.

Очевидно, что общая эффективность параллельного алгоритма глобальной оптимизации зависит как от используемой схемы решения задачи ГО, так и от применяемой стратегии её реализации на высокопроизводительной ЭВМ. В частности, в данной работе в качестве концептуальной идеи повышения эффективности методов решения задачи ГО предлагалось использовать частичное замещение точных методов глобальной оптимизации, имеющих экспоненциальный рост сложности, методами локальной оптимизации, сходящимися к решению за полиномиальное время.

Что же касается реализации выбранной стратегии на высокопроизводительной ЭВМ, то здесь возможны самые разнообразные решения. В работе рассматривались 2 варианта реализации алгоритма ДАМПД. На самом деле, с учетом методов организации обмена данными между процессорами, их значительно больше. Причем невозможно заранее установить, какой из вариантов параллельного алгоритма окажется самым эффективным. В таких условиях большое значение приобретают средства автоматизации моделирования параллельных алгоритмов, к числу которых относится и PGRAPH. Следует иметь в виду, что на рисунках 3 и 6 показаны не схемы параллельных алгоритмов, а готовые программы, представленные в визуальной нотации, которые автоматически компилируются в тексты программ на языке C++, включая директивы MPI, синхронизацию и прочее.

Литература

1. Евтушенко Ю.Г., Посыпкин М.А. Параллельные методы решения задач глобальной оптимизации // PACO'2008: Труды четвертой международной конференции «Параллельные вычисления и задачи управления». М., 2008. С. 5 – 15
2. Квасов Д.Е., Сергеев Я.Д. Многомерный алгоритм глобальной оптимизации на основе адаптивных диагональных кривых // ЖВМ и МФ, 2003, Т.43, № 1. С. 42-59.
3. Коварцев А.Н., Попова-Коварцева Д.А. К вопросу об эффективности параллельных алгоритмов глобальной оптимизации функций многих переменных // Компьютерная оптика. 2011. Т. 35, № 2. С. 256 – 262.
4. Коварцев А.Н., Попова-Коварцева Д.А. Многомерный параллельный алгоритм глобальной оптимизации модифицированным методом половинных делений // В мире научных открытий. № 8.1(32), 2012. С. 80-108.
5. Коварцев А.Н., Жидченко В.В. Моделирование синхронных параллельных вычислений при построении математических моделей сложных систем // Труды первой международной

конференции: Системный анализ и информационные технологии. Том 2. М.: КомКнига, 2005. С.154-160.

6. Посыпкин М.А. Методы решения задач конечномерной оптимизации в распределенной вычислительной среде // САИТ-2009: Труды конференции. М., 2009. С. 729-740
7. Gaviano M., Kvasov D.E., Lera D., Sergeyev Ya.D. Software for generation of classes of test of functions with known local and global minima for global optimization // ASM Transactions on Mathematical Software, 2003. 29(4). pp. 469-480.