ТРЕХУРОВНЕВЫЙ ВАРИАНТ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА МЕТОДА НЕРАВНОМЕРНЫХ ПОКРЫТИЙ ДЛЯ ЗАДАЧ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

В.У. Малкова

На основе метода неравномерных покрытий, предложенного Ю.Г. Евтушенко в [1], был разработан и реализован параллельный алгоритм поиска глобального экстремума липшицевых функций [2, 3]. Случай липшицевых функций является наиболее проработанным. Если известна константа Липшица, то метод неравномерных покрытий дает гарантированное решение, но на практике им удается решить лишь задачи невысокой размерности. Эффективность метода существенно зависит от константы Липшица, которая обычно априори неизвестна и приходится пользоваться ее завышенными верхними оценками, что существенно усложняет расчеты. Вместе с тем, во многих задачах существуют подобласти, где константа Липшица невелика и ориентация на максимальную константу неоправданна.

Причем сложность решения многоэкстремальных задач глобальной оптимизации зависит не только от количества локальных оптимумом функции, но и от их взаимного расположения. Если поведение функции характеризуется большим числом узких «воронок» с крутыми склонами, то процесс поиска может так и не найти глобальный минимум, надолго «застревая» в очередной воронке. Для преодоления этой трудности было сделано предположение, что если начинать покрытие не с одного параллелепипеда, а с их набора, которым отвечают локальные минимумы различных воронок [4], то потребуется намного меньше циклов, чтобы найти глобальный оптимум.

Для реализации этой идеи был предложен параллельный алгоритм, основанный на популяционном подходе [5], который предполагает, что множество кандидатов-решений одной задачи поддерживается в семействе, эволюционирующем таким образом, что гарантируется достаточная диверсификация решений. Обеспечение такого разнообразия достигается благодаря введению меры различия между решениями, которая зависит от специфики решаемой задачи. Проведенные вычислительные эксперименты показали эффективность этого алгоритма, позволившего существенно сократить время поиска оптимума.

Однако с увеличением числа доступных процессоров в современных суперкомпьютерных центрах объем обмена сообщениями между процессором-диспетчером и рабочими процессорами может существенно возрасти, так что рабочие процессоры будут немалую долю рабочего времени проводить в очередях, в ожидании отклика диспетчера. Была предложена новая схема организации взаимодействия между процессорами, а именно, введен промежуточный уровень – процессоров-бригадиров, которые берут на себя часть функций диспетчера. Оставшиеся рабочие процессоры динамически разбиваются по бригадам, при этом каждый рабочий процессор взаимодействует со своим бригадиром в асинхронном режиме, тем самым устраняется «узкое место» обмена сообщениями с диспетчером. Бригадиры, со своей стороны, взаимодействуют с диспетчером и своими рабочими процессорами.

Трехуровневый параллельный алгоритм с множеством кандидатов-решений

Идея параллельного алгоритма состоит в том, чтобы выполнять итерации последовательного метода параллельно на множестве процессоров с периодическим обменом рекордами и перераспределением областей поиска между ними в процессе счета. При выполнении итераций последовательного алгоритма каждый рабочий процессор поддерживает свой индивидуальный пул параллелепипедов. Будем говорить, что наступает завершение работы процессора, если им выполнены Q циклов последовательного алгоритма или если его индивидуальный пул стал пустым. Процессор, который завершил работу, сообщает диспетчеру следующие величины: число параллелепипедов в своем пуле, минимальную оценку и свой индивидуальный рекорд. На основе полученных данных диспетчер улучшает рекорд и сообщает его всем ожидающим процессорам, которые будут использовать новый рекорд для отсева параллелепипедов из своих наборов.

Рабочие процессоры выполняют итерации алгоритма метода неравномерных покрытий, при этом каждый процессор ведет свое семейство кандидатов-решений согласно выбранной мере различия. В идеале мера различия d должна быть такова, что если точки X и Y принадлежат одной воронке (т.е. траектория движения из X и Y приводит к дну одной и той же воронки), то $d(X,Y) \le d_0$, где d_0 – пороговое значение. Любая новая точка Y становится кандидатом на включение в семейство. В качестве новых точек служат центры параллелепипедов, полученные в процессе половинных делений, и точки, найденные в результате локального поиска экстремума. Для обновления семейства используется следующий алгоритм:

1. Среди членов семейства S ищется точка X_q, наиболее «близкая» к новой точке Y в соответствии с выбранной мерой различия;
2. Если мера различия $d(X_q, Y) \le d_0$ и при этом $f(X_q) > f(Y)$, то точку X_q заменяем на Y;
3. Если $d > d_0$ и число членов семейства меньше предельного значения M, то точка Y включается в семейство в качестве нового члена;

4. Если $d > d_0$ и число членов семейства равно *M*, то в семействе ищется точка X_p с наихудшим значением функции *f*. Если $f(X_p) > f(Y)$, то точку X_p заменяем на точку Y.

Предлагается следующий трехуровневый алгоритм организации взаимодействия между процессорами: пусть имеется начальное семейство из *M* членов; число процессоров *P*, число процессоров-бригадиров *T*.

Алгоритм для процессора-диспетчера:

1. Инициализация: процессоры с номерами от *1* до *T* считаются бригадирами, а остальные – свободными рабочими процессорами в режиме ожидания. Диспетчер ждет сообщений от бригадиров.
2. Диспетчер просматривает список бригадиров, ожидающих команды. Если у бригадира с номером *J* непустой пул и имеется хотя бы один свободный рабочий процессор с номером *I*, то диспетчер дает бригадиру приказ зачислить процессор *I* в свою бригаду (команда «enroll(*I*)»), а рабочему – поступить в бригаду *J* (команда «join (*J*)»). Далее этот рабочий и бригадир взаимодействуют между собой.
3. Если у бригадира с номером *J* непустой пул, но свободных рабочих процессоров нет, то ему дается приказ продолжать работу (команда «work»).
4. Если у бригадира с номером *J* пустой пул и есть хотя бы один ожидающий бригадир с номером *L*, у которого в пуле более одного параллелепипеда, то бригадиру *J* дается приказ взять параллелепипед у бригадира *L* (команда «take(*L*)»), а бригадиру *L* – приказ отдать параллелепипед бригадиру *J* (команда «give(*J*)»).
5. Если пулы у всех бригадиров пусты или истекло заданное время, то всем процессорам дается команда «finish». От всех процессоров принимаются семейства и сводятся в одно итоговое семейство кандидатов-решений. Работа завершается.

Алгоритм для процессора-бригадира с номером J:

1. Инициализация: бригадир берет точку с соответствующим номером *J* из начального семейства кандидатов-решений и строит вокруг нее область поиска – параллелепипед с заданной длиной ребра. Бригадир сообщает диспетчеру, что у него имеется 1 параллелепипед, значение своего рекорда и оценки, после чего ждет приказа диспетчера.
2. Если бригадир получает приказ зачислить процессор с номером *I* в свою бригаду, то он ждет «явки» процессора *I*.
3. Если бригадир получает от процессора *I* сообщение, что пул у него пуст, то он просматривает список ожидающих процессоров, входящих в его бригаду. Если среди них есть рабочий процессор *L*, в пуле которого более одного параллелепипеда, то бригадир дает ему приказ отдать параллелепипед процессору *I* (команда «give(*I*)»), а процессору L – принять параллелепипед (команда «take(*L*)»).
4. Всем процессорам, у которых в пуле один параллелепипед, дается приказ продолжать работу (команда «work»).
5. Если в списке ожидающих нет свободных процессоров, то всем рабочим процессорам дается команда «work».
6. Если все члены бригады свободны, то бригадир сообщает диспетчеру, что он простаивает и ждет приказа.
7. Если бригадир получил от диспетчера команду «finish», то он дает команду «finish» всем членам своей бригады, а затем передает диспетчеру свое семейство кандидатов-решений и завершает работу.

Алгоритм для рабочего процессора с номером I:

1. Инициализация: рабочий процессор ждет приказа от диспетчера.
2. Если процессор получает от диспетчера приказ поступить в бригаду с номером *J* (команда «join (*J*)»), то он сообщает бригадиру *J* о том, что он явился и свободен (пул у него пуст).
3. Если процессор получает от бригадира приказ взять или отдать параллелепипед, то он выполняет его и сообщает бригадиру свой рекорд и число параллелепипедов в пуле.
4. Если процессор получает от бригадира приказ работать (команда «work»), то он выполняет *Q* итераций последовательного алгоритма метода неравномерных покрытий, после чего сообщает бригадиру свой рекорд и число параллелепипедов в пуле.
5. Если процессор получает от бригадира приказ команду «finish», то он передает диспетчеру свое семейство кандидатов-решений и завершает работу.

Алгоритм реализован на языке С в MPI-системе параллельного программирования. Для ускорения расчетов используются вспомогательные процедуры поиска локального экстремума. Эффективность предложенного алгоритма проверена на тестовых задачах. Вычислительные эксперименты, выполненные на многомашинном вычислительном комплексе MVS100K Межведомственного суперкомпьютерного центра РАН, подтвердили реальную возможность поиска глобальных решений.

ЛИТЕРАТУРА:

1. Ю.Г. Евтушенко. "Численный метод поиска глобального экстремума функций (перебор на неравномерной сетке)" // М.: ЖВМ и МФ, 1971. 11, №6, с.1390-1403.

2. Ю.Г. Евтушенко, В.У. Малкова, А.А. Станевичюс. "Параллельный поиск глобального экстремума функции многих переменных" //Ж. вычисл. матем. и матем. физ., 2009, том 49, №2, с. 255 - 296.
3. Ю.Г. Евтушенко, В.У. Малкова, А.А. Станевичюс. "Распараллеливание процесса поиска глобального экстремума" //Автоматика и телемеханика, №5, 2007, с. 46- 58.
4. A. Grosso., M. Locatelli, F. Schoen. "A population-based approach for hard global optimization problems based on dissimilarity measures" //Math. Program., Ser. A (2007), pp. 373–404.
5. Ю.Г. Евтушенко, В.У. Малкова, А.А. Станевичюс. "Параллельный метод неравномерных покрытий с множеством кандидатов-решений". Материалы Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ", Новороссийск, 21-26 сентября 2009 г., Изд-во Московского Университета, с.91-97.