Алгоритм решения нестационарных задач линейного программирования для кластерных вычислительных систем с многоядерными ускорителями

И.М. Соколинская, Л.Б. Соколинский

Южно-Уральский государственный университет

В работе предлагается новый алгоритм для решения нестационарных задач линейного программирования большой размерности, ориентированный на кластерные вычислительные системы с многоядерными ускорителями. В основе алгоритма лежат фейеровские отображения. Алгоритм отслеживает изменения исходных данных и вносит корректировки в вычислительный процесс. При этом задача разбивается на большое количество подзадач, которые могут решаться независимо без обменов данными. Эффективное использование многоядерных ускорителей обеспечивается применением метода деления вектора на подвекторы.

1. Введение

В практике экономико-математического моделирования часто встречаются задачи линейного программирования большой размерности с быстро меняющимися входными данными. Одним из примеров является задача управления портфелем ценных бумаг с использованием методов алгоритмической торговли [1,2]. В подобных задачах количество переменных и неравенств в системе ограничений может составлять десятки и даже сотни тысяч, а период изменения исходных данных находиться в пределах сотых долей секунды. При решении подобных задач симплекс-метод и метод опорных векторов оказываются неэффективными, так как исходные данные меняются быстрее, чем алгоритм заканчивает работу. Для преодоления этой проблемы предлагается новый *«следящий»* алгоритм решения задачи линейного программирования с использованием фейеровских отображений [3], ориентированный на кластерные вычислительные системы с многоядерными ускорителями. Статья организована следующим образом. В разделе 2 делается математическая постановка задачи линейного программирования, даются определения фейеровского отображения и операции псевдопроектирования на многогранник. В разделе 3 описывается новый параллельный алгоритм решения нестационарной задачи линейного программирования, получивший название *«следящий»*. В разделе 4 предлагается модель, позволяющая имитировать нестационарность входных данных, что является полезным при исследовании следящего алгоритма. В заключении суммируются полученные результаты и определяются направления дальнейших исследований.

2. Математическая постановка задачи

Пусть задана задача линейного программирования

$$\max\left\{\langle c,x\rangle \,|Ax\le b,\ x\ge 0\right\}. \tag{1}$$

Положим $l_i\left(x\right)=\langle a_i,x\rangle-b_i$. Определим фейеровское отображение следующим образом:

$$\varphi\left(x\right)=x-\left(\frac{\lambda}{\delta}\right)\sum_{i=1}^{m}l_i^{+}\left(x\right)a_i.$$

Здесь l^{+} – положительная срезка: $l^{+}=\max\left(l,0\right)$; $\delta=\sum\limits_{i=1}^{m}\left\|a_i\right\|^2$.

Пусть M – многогранник, задаваемый ограничениями задачи линейного программирования (1). Такой многогранник всегда является выпуклым. Обозначим

$$\varphi^s(x) = \underbrace{\varphi \dots \varphi}_{s}(x).$$

Под *фейеровским процессом*, порождаемым отображением φ при произвольном начальном приближении $x_0 \in \mathbb{R}^n$, будем понимать последовательность $\{\varphi^s(x_0)\}_{s=0}^{+\infty}$. Известно, что указанный фейеровский процесс сходится к точке, принадлежащей множеству M:

$$\{\varphi^s(x_0)\}_{s=0}^{+\infty} \to \bar{x} \in M.$$

Будем кратко обозначать это следующим образом: $\lim\limits_{s\to\infty} \varphi^s(x_0) = \bar{x}$.

Под *φ-проектированием* (*псевдопроектированием*) точки $x \in \mathbb{R}^n$ на многогранник M понимается отображение $\pi_M^{\varphi} : \mathbb{R}^n \to M$ [4], задаваемое соотношением $\pi_M^{\varphi}(x) = \lim\limits_{s\to\infty} \varphi^s(x)$.

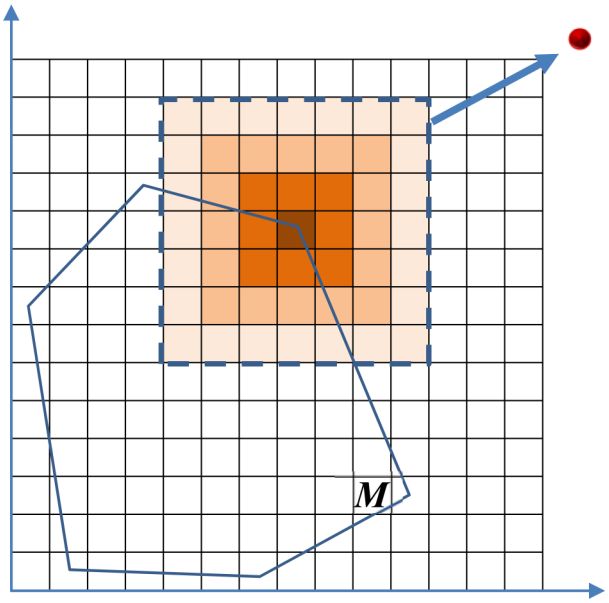

Рис. 1. Следящая область кубической формы с целевой ячейкой в центре.

3. Параллельный алгоритм

В общем виде следящий алгоритм может быть описан следующим образом (см. Рис. 1). Все пространство $\mathbb{R}^n$ делится гиперкубической сеткой на ячейки с переменной длиной грани. Размер ячейки может динамически меняться в зависимости от скорости изменения исходных данных: чем быстрее меняются данные, тем больше становится ячейка, чтобы успевать следить. Алгоритм постоянно находит (отслеживает) ячейку наименьшего размера, на которой достигается максимум целевой функции. Будем такую ячейку называть целевой. В каждой итерации алгоритма просчитываются ячейки, входящие в некоторую следящую область вокруг целевой ячейки. В простейшем случае следящая область может быть кубической формы со следящей ячейкой в центре. В общем случае следящая область может быть произвольной выпуклой фигурой. При этом ячейки всегда имеют кубическую форму. *Целевая точка* – это некоторая точка, находящаяся вне следящей области. Ее координаты могут динамически вычисляться по коэффициентам целевой функции. Например, при целевой функции $Q(x) = x_1 + 2x_2$ в качестве целевой точки можно взять точку $(T, 2T)$, где T – достаточно большое положительное число. В рамках одной итерации алгоритма выполняются следующие действия.

1. Выбирается кубическая следящая область с длиной диагонали $r = \sqrt{2} \cdot a$ (a – длина грани куба) с центром в точке q, заведомо покрывающая многогранник.

2. Выбирается целевая точка $z = Tc$, расположенная вне следящей области.

3. Следящая область разбивается на K ячеек.

4. В условиях динамического изменения входных данных (A, b, c), для всех ячеек внутри следящей области вычисляется псевдопроекция из точки z на пересечение ячейки с многогранником. Если пересечение пусто, то такие ячейки отбрасываются.

5. Если получено пустое множество псевдопроекций, то мы увеличиваем диагональ следящей области в W раз и переходим на шаг 2.

6. Если получено непустое множество псевдопроекций, то на нем вычисляется максимум целевой функции.

7. Если расстояние от точки максимума до центра следящей области меньше $\frac{1}{2}r$, то длина диагонали следящей области уменьшается на $\frac{1}{4}$.

8. Если расстояние от точки максимума до центра следящей области больше $\frac{1}{2}r$, то длина диагонали следящей области увеличивается на $\frac{1}{4}$.

9. Центр следящей области перемещается в центр ячейки, в которой достигнут максимум, и мы переходим на шаг 2.

Псевдопроекции на шаге 4 для различных ячеек следящей области могут вычисляться параллельно без обменов данными между MPI-процессами. Фейеровский процесс вычисления псевдопроекции, стартовавший из целевой точки на шаге 4, вычисляется до конца, несмотря на то, что координаты целевой точки и сам многогранник в процессе счета могли измениться. Выполнение условия шага 5 означает, что входные данные изменяются быстрее, чем мы вычисляем псевдопроекции. Константы 1/2 и 1/4 , используемые при выполнении шагов 7 и 8, являются параметрами алгоритма.

Эффективное использование многоядерных ускорителей достигается путем распараллеливания процесса вычисления отдельной псевдопроекции, выполняемого на шаге 4. При этом используется метод разбиения вектора на подвекторы [4]. Суть метода заключается в том, что вектор исходной точки делится на подвекторы по числу процессорных ядер многоядерного ускорителя. На каждом подвекторе независимо делается v фейеровских приближений с использованием редуцированных фейеровских отображений. Такое редуцированное отображение воздействует только на «свой» подвектор, оставляя оставшуюся часть вектора без изменений. Затем нити управления через общую память обмениваются модифицированными подвекторами и процесс продолжается. В работе [4] была доказана сходимость описанного итерационного процесса.

Распараллеливание, таким образом, осуществляется на двух уровнях:

- межузловое распараллеливание;
- внутриузловое распараллеливание.

Межузловое распараллеливание – это распараллеливание вычислений между отдельными узлами кластерной вычислительной системы. В качестве технологии параллельного программирования используется MPI. Каждый MPI-процесс работает на отдельном узле кластера. Обмен данными между MPI-процессами осуществляется с помощью передачи сообщений по коммуникационной сети. Один MPI-процесс считает одну псевдопроекцию на пересечение одной ячейки следящей области с многогранником. Таким образом, количество ячеек следящей области всегда равно константе K, совпадающей с количеством MPI-процессов.

Внутриузловое распараллеливание – это распараллеливание вычислений между отдельными процессорными ядрами вычислительного узла. В качестве технологии параллельного программирования используется OpenMP. Обмен данными между нитями (потоками управления) осуществляется через общую память. Объектом распараллеливания на этом уровне является отдельный фейеровский процесс, вычисляющий псевдопроекцию. В основе распараллеливания лежит метод деления вектора на подвекторы. Для каждого подвектора организуется отдельная нить, вычисляющая фейеровский процесс на подвекторе.

4. Моделирование нестационарности входных данных

Под нестационарностью входных данных понимается динамическое (происходящее в ходе выполнения алгоритма) изменение всех или некоторых элементов матрицы A и векторов b и c из формулировки задачи линейного программирования (1). Для исследования следящего алгоритма нами была разработана следующая модель, имитирующая поведение нестационарных входных данных.

Различаются *нулевые* и *ненулевые элементы*. *Ненулевой элемент* может принимать произвольное значение в разные моменты времени (отрицательное, положительное, равное нулю). *Нулевой элемент* всегда равен нулю. Все ненулевые элементы будем называть *динамическими параметрами* задачи (1). Нестационарные характеристики динамического параметра α определяются функцией времени $f_\alpha(t)$, $f_\alpha : \mathbb{R} \to \mathfrak{D}_\alpha$, где $\mathfrak{D}_\alpha$ – множество всех возможных значений параметра α. Будем называть эту функцию *функцией параметра*. На функцию параметра могут накладываться следующие ограничения:

- непрерывность;
- гладкость.

С каждым динамическим параметром α связывается константа $\delta \in \mathbb{R}_{\geqslant 0}$, называемая *отклонением*. Для отклонения δ на интервале $(-\delta, \delta)$ задается случайная величина X с равномерной функцией распределения $F(\chi) = \frac{\chi+\delta}{2\delta}$. Значение параметра α определяется по формуле $\alpha = f_\alpha(t) + \mathrm{X}$.

В упрощенном виде функцию $f_\alpha(t)$ можно определить следующим образом $f_\alpha(t) = \sigma \sin(\omega t + \xi) + \gamma$, где $\gamma \in \mathbb{R}$, $\sigma, \omega \in \mathbb{R}_{\geqslant 0}$, $\xi \in [0, 2\pi/\omega)$. Положим $\delta = \frac{\sigma}{D}$, где D – некоторое натуральное число. Тогда произвольный динамический параметр α задается четверкой $(\sigma, \omega, \xi, \gamma)$. При этом $(0, \omega, \xi, \gamma)$ соответствует параметру-константе со значением γ, которое не меняется во времен; $(0, \omega, \xi, 0)$ соответствует нулевому параметру. За единицу времени можно выбрать время вычисления одного значения фейеровского отображения. При этом параметр ω будет ускорять течение времени при значениях $\omega > 1$, либо замедлять при значениях $\omega < 1$.

5. Заключение

В работе был предложен новый параллельный «следящий» алгоритм для решения нестационарных задач большой размерности на кластерных вычислительных системах с многоядерными ускорителями. Распараллеливание в алгоритме осуществляется на двух уровнях: межузловом и внутриузловом. Межузловой уровень не предполагает обменов данными по соединительной сети и поэтому обеспечивает практически линейную масштабируемость алгоритма на вычислительных системах с сотнями тысяч вычислительных узлов. Внутриузловое распараллеливание допускает эффективное использование процессорных ядер многоядерных ускорителей. С целью изучения нового алгоритма была разработана параметризованная математическая модель, имитирующая нестационарность входных данных. Авторы планируют реализовать описанный алгоритм на языке Си с использованием технологий параллельного программирования MPI и OpenMP. Также будет реализована

система имитации нестационарности входных данных, с помощью которой предполагается произвести экспериментальное исследование следящего алгоритма и оценить его эффективность применительно к задачам линейного программирования большой размерности.

Литература

1. *Дышаев М.М., Соколинская И.М.* Представление торговых сигналов на основе адаптивной скользящей средней Кауфмана в виде системы линейных неравенств // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2013. Т. 2. № 4. С. 103-108.

2. *Ананченко И.В., Мусаев А.А.* Торговые роботы и управление в хаотических средах: обзор и критический анализ // Труды СПИИРАН. 2014. № 3 (34). С. 178-203.

3. *Еремин И.И.* Фейеровские методы для задач выпуклой и линейной оптимизации. Челябинск: Изд-во ЮУрГУ, 2009. 200 с.

4. *Ершова А.В., Соколинская И.М.* О сходимости масштабируемого алгоритма построения псевдопроекции на выпуклое замкнутое множество // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2012. № 18 (277). С. 5-12.