УДК 519.853.4

Б.О. ОНИЩЕНКО

РЕШЕНИЕ ОДНОЙ ЗАДАЧИ СТОХАСТИЧЕСКОЙ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ ПАРАЛЛЕЛЬНЫМ МЕТОДОМ ВЕТВЕЙ И ГРАНИЦ НА КЛАСТЕРЕ

Рассматривается задача об оптимальном размещении сервисных центров. Для ее решения предлагается стохастический метод ветвей и границ. Также описан способ распараллеливания предлагаемого алгоритма для работы на кластерных системах. Представлены результаты численных экспериментов.

Введение. В работе рассматривается задача оптимального размещения сервисных центров, которые обслуживают потребителей, расположенных на некоторой географической территории. Эта задача является одной из базовых задач исследования операций. В качестве сервисных центров могут быть магазины, склады, аварийные службы, операторы мобильной связи, станции скорой помощи, заправочные станции и прочие поставщики услуг, потребителями – население и прочие мелкие потребители товаров и услуг. Следовательно, задача оптимального размещения сравнительно небольшого числа поставщиков для обслуживания большого количества потребителей имеет достаточно широкую область применения.

Предположим, что потребители некоторой услуги неравномерно распределены в некоторой области на плоскости (или вдоль некоторой линии). Потребность в получении услуги клиентом из определенной географической точки может быть случайной. Пусть стоимость обслуживания клиента, расположенного в определенной точке области, в сервисном центре, который расположен в некоторой другой точке области, задается известной функцией, зависящей от расстояния между этими точками. Если есть несколько сервисных центров, то клиент выбирает ближайший к нему центр, возможно, с некоторой вероятностью. Задача состоит в расположении этих центров так, чтобы сово-

купные ожидаемые затраты на обслуживание всех потребителей услуги были минимальными.

Приведем формальную постановку вышеописанной задачи [1]. Пусть потребители некоторой услуги распределены в области $\Omega \subset R^m$, согласно распределения $P(d\omega)$. Стоимость обслуживания клиента, расположенного в точке $\omega \in \Omega$ в сервисном центре, который размещен в точке $x_i \in X \subset R^m$, задается функцией $c(x_i,\omega)$, например, $c(x_i,\omega) = c\left(\|x_i - \omega\|\right)$, в частности,

$$c(x_i,\omega) = \frac{\|x_i - \omega\|^\alpha}{\gamma + \|x_i - \omega\|^\beta},\ \alpha \ge \beta > 0,\ \ \gamma > 0. \qquad (1)$$

Если есть n сервисных центров в точках $x_1,...,x_n \in X$, а клиент выбирает ближайший к нему центр, то стоимость обслуживания клиента ω задается функцией

$$f(x,\omega) = \min_{1\le i\le n} c(x_i,\omega),\ x = (x_1,...,x_n).$$

Задача состоит в размещении n сервисных центров так, чтобы суммарные ожидаемые затраты обслуживания всех потребителей услуги минимизировались:

$$F(x) = \int_\Omega \{\min_{1\le i\le n} c(x_i,\omega)\} P(d\omega) \to \min_{x\in D}, \qquad (2)$$

где D – множество допустимых позиций центров. Координаты центров могут удовлетворять некоторым дополнительным ограничениям, например, они могут быть упорядоченными по отношению к некоторому направлению $d \in R^m$, тогда

$$D = \left\{x = (x_1,...,x_n) \mid x_1 \in X,...,x_n \in X;\ dx_i \le dx_{i+1},\ i = 1,...,n-1\right\}.$$

Данная задача относится к классу многоэкстремальных задач стохастического программирования и поэтому являются весьма сложной для решения.

Для решения поставленной задачи можно использовать стохастический метод ветвей и границ с минорантными оценками значений целевой функции [2].

Построение касательных минорант. Пусть функции стоимости $c(x,\omega)$ обслуживания клиента ω в центре с координатой x допускают касательные миноранты $\phi(x,y,\omega)$. Тогда функции $\varphi(x,y,\omega) = \min_{1\le i\le n} \phi(x_i,y_i,\omega)$ являются касательными минорантами функции минимума $f(x,\omega) = \min_{1\le i\le n} c(x_i,\omega)$, а функции $\Phi(x,y) = \int_\Omega \varphi(x,y,\omega) P(d\omega)$ являются касательными минорантами $F(x)$.

Если $c(x,\omega)$ – гладкая по x функция с липшицевым градиентом, то касательными минорантами $\phi(x,y,\omega)$ могут выступать касательные параболоиды. Заметим, что функция $c(x_i,\omega) = \|x_i - \omega\|^\alpha \Big/ \left(\gamma + \|x_i - \omega\|^\beta\right)$ может быть представлена как разность двух выпуклых функций:

$$c(x,\omega) = \frac{1}{\gamma}\|x - \omega\|^\alpha - \|x - \omega\|^{\alpha+\beta} \Big/ \gamma\left(\gamma + \|x - \omega\|^\beta\right).$$

Поэтому в качестве ее касательной в точке y миноранты можно взять вогнутую функцию

$$\phi(x,y,\omega)=\frac{1}{\gamma}\|y-\omega\|^{\alpha}+\frac{\alpha}{\gamma}\|y-\omega\|^{\alpha-2}\langle(y-\omega),x-y\rangle-\frac{\|x-\omega\|^{\alpha+\beta}}{\gamma\left(\gamma+\|x-\omega\|^{\beta}\right)}.$$

Обоснование необходимости распараллеливания задачи. При размещении достаточно большого количества сервисных центров (т.е. решении задачи большой размерности) возникает ряд осложнений при реализации алгоритма на персональных ЭВМ:

– нехватка оперативной памяти для размещения данных задачи;

– сравнительно небольшая скорость вычислений, что увеличивает время работы программы к недопустимой величине.

В связи с вышеперечисленными трудностями возникает необходимость поиска более эффективного способа реализации алгоритма. Идея метода ветвей и границ состоит в постепенном улучшении верхней и нижней оценок подмножеств, на которые разбивается исходное допустимое множество задачи. Вычисление оценок каждого отдельного подмножества никак не зависит от других подмножеств. Следовательно, после разбития исходного множества на подмножества на каждом из них можно в отдельности и независимо от других решать поставленную задачу, а потом избрать наилучший результат в качестве ответа. То есть методу ветвей и границ присущий естественный параллелизм.

Учитывая то, что метод ветвей и границ может быть естественно распараллеленный, для реализации алгоритма предлагается использовать средства параллельного программирования, а для работы параллельной программы – кластерную систему, как один из видов многопроцессорных систем.

Описание параллельного алгоритма. На этапе начального разбиения допустимого множества на подмножества полученные фрагменты поровну распределяются между узлами кластера, на каждом из которых независимо выполняется некоторое количество итераций метода (поиск и дробление «рекордного» подмножества, удаление бесперспективных подмножеств, уточнение верхних и нижних оценок для подмножеств). После выполнения необходимого числа итераций на всех узлах происходит общее отбрасывание бесперспективных подмножеств, после чего проводится перераспределение остальных подмножеств между узлами кластера.

Для реализации параллельного варианта алгоритма метода ветвей и границ используется библиотека MPI, реализованная под язык программирования Ci.

Алгоритм программы. Программа работает по следующему алгоритму:

1) инициализация MPI;

2) считывание процессом с номером 0 входных данных из файла и их пересылка всем процессам кластера;

3) начальное разбиение допустимого множества на подмножества;

4) распределение полученных подмножеств между процессами;

5) вычисление центра тяжести, верхней и нижней оценок каждым процессом для своих подмножеств;

6) определение подмножества с минимальной оценкой снизу;

7) если не выполняются условия выхода (разность между оценкой сверху и оценкой снизу меньше заданной точности) или условие синхронизации процессов (выполнено определенное количество итераций на каждом процессе), то проводится разбитие найденного подмножества на два новых подмножества и вычисление для них центров тяжести, верхней и нижней оценок, иначе выполняется п. 9;

8) удаление из списка бесперспективных подмножеств и переход к п. 6;

9) если на одном из процессов выполнилось условие выхода из итерационного процесса, а на всех остальных процессах – условие синхронизации, то проводится сравнение нижних оценок «рекордных» подмножеств. Если оценка снизу «рекордного» подмножества в данном процессе является наименьшей, то формируется ответ, сообщение о завершении итерационного процесса и выполняется переход к п. 12;

10) если нет сообщения о завершении итерационного процесса, то происходит удаление из списка бесперспективных подмножеств по наименьшей среди всех процессов оценке сверху;

11) перераспределение подмножеств между процессами и переход к п. 6;

12) процесс, на котором было найдено решение, пересылает полученный результат в процесс с номером 0;

13) процесс с номером 0 выводит координаты оптимального размещения сервисных центров и общее время работы программы.

Структура программы. Программа состоит из следующих функций:

1. `int main(int argc, char* argv[])` – главная функция программы, состоит из
   - блока инициализации MPI;
   - блока считывания и пересылки входных данных;
   - блока разбиения исходного множества задачи на подмножества и их распределения между процессами;
   - основного цикла программы, который состоит из следующих частей:
     - поиск «рекордного» подмножества;
     - проверка условия прекращения итерационного процесса;
     - перераспределение подмножеств между процессами;
     - вызов функции разбиения «рекордного» подмножества на новые подмножества;
   - блока формирования и вывода результатов.

2. `void new_fragm(int i_s)` – функция, которая осуществляет разбиение подмножества с номером `i_s`, а также вызывается функция удаления из списка «бесперспективных» подмножеств.

3. `void set_rebro(int i_s, int nach, int kon, double *rebro)` – используется функцией `new_fragm()` для определения координат

ребра подмножества с номером i_s, образованного вершинами с номерами nach и kon.

4. void del_no_future_fragms(double bound) – функция, которая проводит удаление из списка «бесперспективных» подмножеств, т.е. множеств, нижняя оценка которых больше величины bound.

5. void clear_y() – осуществляет удаление вершин, которые не задействованы ни в одном подмножестве.

6. double set_min_up_o(double nach_min) – используется функцией new_fragm() для определения наименьшей оценки снизу для подмножеств.

7. void del_fragm(int i_s) – удаляет из списка подмножеств подмножество с номером i_s.

8. void set_sr_fragm(int i_s) – вычисляет центр тяжести подмножества с номером i_s.

9. double set_up_o(int i_s) – вычисляет оценку сверху подмножества с номером i_s.

10. double set_down_o(int i_s) – вычисляет оценку снизу подмножества с номером i_s.

11. double norma(double *x) – вычисляет норму вектора x.

12. double c(double xi, double tet) – функция стоимости обслуживания клиента, расположенного в точке с координатой tet, в сервисном центре, расположенном в точке с координатой xi.

13. double dc(double xi, double tet) – значение частной производной функции c по xi.

14. double f(double *x) – целевая функция.

15. double fi(double *x, double *y) – касательная в точке y миноранта к функции f().

16. double minfi_loc(double *x, int n_fr) – используется функцией set_down_o() для вычисления минимума касательных минорант, построенных в вершинах подмножества с номером n_fr.

Программа предназначена для работы на кластерных системах, построенных с использованием узлов на базе процессоров Intel или AMD и управляемых операционной системой Linux и средами, которые поддерживают MPI (например, MPICH или LAM/MPI).

Входные данные для описываемой программы:

– количество сервисных центров, которые необходимо разместить;

– точность вычислений ($\varepsilon > 0$);

– количество клиентов;

– параметр, который определяет стоимость обслуживания клиента в сервисном центре ($\gamma > 0$);

– список координат клиентов и вероятностей локализации клиента в указанной точке.

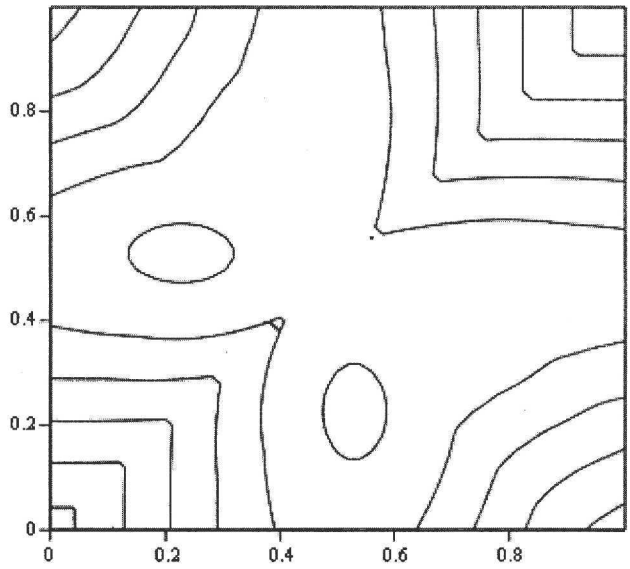
Выходными данными программы являются координаты точек размещения сервисных центров и время работы программы.

Численные эксперименты. Для иллюстрации работы описанного алгоритма и программы были использованы следующие входные данные. В функции (1) $\alpha = \beta = 2$, а $\gamma = 0.1$. Размещение потребителей $\omega_1, ..., \omega_m$ – равномерно распределенные случайные величины на отрезке [0,1] с вероятностями $p_{\omega_1}, ..., p_{\omega_m}$ соответственно. Для $m = 20$ значения ω_i и p_{ω_i} приведены в табл. 1.

ТАБЛИЦА 1. Распределение потребителей

Значение	0.037991	0.100437	0.193362	0.260176	0.326991
Вероятность	2.15658E-02	3.55347E-03	8.26707E-02	9.97986E-02	2.58563E-02
Значение	0.442882	0.453538	0.480874	0.518605	0.518865
Вероятность	1.20332E-01	1.42203E-01	9.61612E-02	7.31521E-02	2.01054E-02
Значение	0.595109	0.606194	0.646899	0.720796	0.747336
Вероятность	7.57130E-02	8.86943E-02	4.32473E-02	1.66506E-02	3.97192E-02
Значение	0.835398	0.847773	0.885398	0.933185	0.999739
Вероятность	2.55506E-02	1.52854E-02	8.41436E-03	1.32620E-03	7.08675E-08

На рисунке показано изображение линий уровня функции (2) при размещении двух сервисных центров. Очевидно, что целевая функция является многоэкстремальной функцией со сложным рельефом.



РИСУНОК

Тестирование программы проводилось на 32-процессорном кластерном комплексе, который смонтирован в Институте кибернетики им. В.М. Глушкова НАН Украины. Работа кластерного комплекса осуществляется под управлением операционной системы Linux, средами MPICH и LAM/MPI. Результаты работы программы приведены в табл. 2.

ТАБЛИЦА 2. Результаты работы программы

Количество сервисных центров	X^*
1	(0.483887)
2	(0.227458, 0.529541)
3	(0.657471, 0.468994, 0.227295)
4	(0.227539; 0.469727; 0.610352; 0.790527)

В табл. 3 приведено время работы программы по размещению четырех сервисных центров между двадцатью клиентами, в зависимости от количества задействованных процессоров. Очевидно, что время работы программы уменьшается с увеличением количества процессоров, причем больше чем линейно. Это происходит за счет малого количества междупроцессорноых взаимодействий, а также за счет эффективного отбрасывания бесперспективных подмножеств.

ТАБЛИЦА 3. Время работы программы по отношению к количеству процессоров

Количество			Приблизительное время работы программы
клиентов	сервисных центров	процессоров	
20	4	1 (ПК: Pentium-4, 1,6 Гц)	≈3,5 ч
20	4	4	≈35 мин
20	4	10	≈15 мин

Заключение. В работе описан новый оригинальный метод решения многоэкстремальних задач стохастической оптимизации – стохастический метод ветвей и границ с минорантными оценками ветвей для решения задачи оптимального размещения сервисных центров. Стохастический метод ветвей и границ является обобщением на стохастические задачи известного детерминированного метода ветвей и границ, который широко используется для решения задач детерминированной дискретной многоэкстремальной оптимизации [3]. Параллельная реализация стохастического метода ветвей и границ позволит значительно ускорить решение задачи, теоретически, соразмерно числу использованных процессоров, а также увеличить размерность решаемой задачи.

Б.О. Онищенко

РОЗВ'ЯЗАННЯ ОДНІЄЇ ЗАДАЧІ СТОХАСТИЧНОЇ ГЛОБАЛЬНОІ ОПТИМІЗАЦІЇ ПАРАЛЕЛЬНИМ МЕТОДОМ ГІЛОК І ГРАНИЦЬ НА КЛАСТЕРІ

Розглядається задача про оптимальне розміщення сервісних центрів. Для знаходження її розв'язку пропонується стохастичний метод гілок і границь. Також описаний спосіб розпаралелювання запропонованого алгоритму для роботи на кластерних системах. Представлені результати чисельних експериментів.

B.O. Onischenko

SOLVING ONE PROBLEM OF STOCHASTIC GLOBAL OPTIMIZATION OF PARALLEL BRANCH AND BOUNDS METHOD ON THE CLUSTER

In the paper a problem of optimal facility location is considered. Stochastic method of the branch and bound is offered for its decisions. Also the technique of parallelization of suggested algorithm for work on the cluster systems is description. Results of numerical experiments are presented.

1. *Norkin V., Pflug G.Ch., Ruszczynski A.* A branch and bound method for stochastic global optimization // Math. Progr. – 1998. – **83**. – P. 425–450.
2. *Норкин В.И., Онищенко Б.О.* Метод ветвей и границ с минорантными оценками для решения задач стохастической глобальной оптимизации // Компьютерная математика. – Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2004. – №. 1. – С. 91–101.
3. *Сергиенко И.В.* Математические модели и методы решения задач дискретной оптимизации: 2-е изд. доп. и перераб. – Киев: Наук. думка, 1988. – 472 с.

Получено 21.12.2004

Об авторе:

Онищенко Борис Олегович,
аспирант Института кибернетики им. В.М. Глушкова НАН Украины.
e-mail: boris@cdu.edu.ua