УДК 519.832.3

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МЕТОДОВ РЕШЕНИЯ КОНФЛИКТНЫХ ЗАДАЧ

Красников К.Е.

МИРЭА — Российский технологический университет, 119454, Россия, г. Москва, проспект Вернадского, 78, e-mail: krasnikov_k@mirea.ru

Данная работа посвящена программной реализации численных методов поиска решений конфликтных (теоретико-игровых) задач. В качестве основного инструмента решения таких задач выбрана система конфликтных равновесий, разработанная Э.Р. Смольяковым. Приводятся алгоритмы решения задач с платёжными функциями участников, заданными в виде матриц, которыми далее аппроксимируются задачи с непрерывными функциями. Для увеличения производительности алгоритмов осуществляется их адаптация под архитектуру параллельных вычислений на процессорах видеокарты (GPU) фирмы nVidea CUDA.

Ключевые слова: теория игр, оптимальное управление, параллельные вычисления, GPU, CUDA.

SOFTWARE IMPLEMENTATION OF METHODS FOR SOLVING CONFLICT PROBLEMS

Krasnikov K.E.

Russian Technological University, 119454, Moscow, 78 Vernadskogo Avenue, Russia, e-mail: krasnikov_k@mirea.ru

The article is devoted to the software implementation of numerical methods for finding solutions to conflict (game-theoretic) problems. The system of conflict equilibria developed by E.R. Smolyakov was chosen as the main tool for solving such problems. Algorithms for solving problems with utility functions of participants are given in the form of matrices and continuous functions. In order to increase the performance of the algorithms, they are adapted to the architecture of parallel computing nVidia CUDA.

Keywords: game theory, optimal control, parallel computing, GPU, CUDA.

Введение

Как известно, математическая теория игр является довольно распространённым инструментом решения задач, целью которых является выбор оптимальной стратегии поведения рационально мыслящих участников в условиях конфликтной ситуации в любой из возможных сфер человеческой деятельности: экономике, военном деле, биологии, социологии, психологии и т.д. [3]

Однако, найти аналитическое решение, удовлетворяющее всем требованиям постановки, возможно далеко не во всех конфликтных задачах.

В этой связи большой практический интерес представляют численные методы, позволяющие найти приближённое решение задач с любой заданной точностью, и программная реализация этих методов.

Целью работы является разработка и программная реализация алгоритмов решения конфликтных задач.

Для выполнения данной цели необходимо решить следующие задачи:

1. Разработать численные методы и их программную реализацию решения конфликтных задач с двумя участниками, платёжные функции которых заданы в виде матриц.
2. Произвести аппроксимации непрерывных платёжных функций участников, матричными функциями.
3. Произвести оптимизацию разработанных алгоритмов решения конфликтных задач под архитектуру параллельных вычислений на процессорах видеокарты (GPU) компании nVidea CUDA.

Постановка конфликтной задачи с двумя участниками

Будем рассматривать конфликтные задачи, удовлетворяющие следующим допущениям.

Допущение 1. *Пусть* $Q_i, i = 1,2$ *– координатные оси двумерного евклидового пространства* $Q = Q_1 \times Q_2; q = (q_1, q_2)$ *– произвольная точка на плоскости* Q*;* G *– произвольное замкнутое ограниченное множество в* Q*;* $G(q_i), i = 1,2$ *– сечения множества* G*, проходящие через заданную в* G *точку* $q = (q_1, q_2)$*;* $Pr_{Q_i}G$ *– проекция*

множества G на ось Q_i; $J_i(q_1, q_2)$, $i = 1,2$ – вещественные ограниченные функции, определённые на G, непрерывные по каждой переменной в отдельности при всех допустимых фиксированных значениях другой.

i-й участник, выбирая стратегию (точку) q_i доступного ему сечения $G(q_k)$ $(k \neq i, i = 1,2)$ множества G или из проекции $Pr_{Q_i}G$ множества G на ось Q_i, стремится обеспечить максимум своей платёжной функции (функционала) $J_i(q)$, $i = 1,2$, причём допустим только такой выбор стратегий игроками, при котором реализующаяся точка (ситуация) q принадлежит множеству G.

Целью задачи является найти те точки игрового множества G, которые можно считать *равновесными* в том смысле, что игроки в силу определённых причин не пожелали бы изменить свои стратегии, составляющие равновесные игровые ситуации.

Одним из наиболее широко используемых является равновесие, определенное Дж. Ф. Нэшэм в 1951 году и носящее его имя [5].

Определение 1. *Ситуацию (точку) $q^* \in G$ назовём равновесной по Роусу-Нэшу* (или C^N-экстремальной), если

$$\max_{q_i \in G(q_k^*)} J_i(q_i, q_k^*) = J_i(q^*), \qquad i = 1,2; k \neq i$$

Однако данное ставшее классическим понятие конфликтного равновесие при всей своей интуитивной простоте имеет ряд весьма существенных недостатков.

Во-первых, оно не всегда существует.

Во-вторых, как это хорошо иллюстрирует ставшая хрестоматийной задача о Дилемме заключённого [4], даже в случае существование, удовлетворяющая данному определению игровая ситуация может оказаться абсолютно невыгодной причём сразу для всех участников.

Перечисленные недостатки побудили исследователей искать новые понятия конфликтных равновесий, способных преодолеть перечисленные выше недостатки.

Весьма успешных результатов в этом направлении удалось добиться Э.Р. Смольякову, разработавшему систему постепенно усиливающихся конфликтных равновесий [6].

Приведём определение базового понятия равновесия данной системы.

Определение 2. *Точку (ситуацию) $q^* \in G$ назовём A_i-экстремальной, если при заданной стратегии $q_k^*, k \neq i, k = 1,2$, допустимой оказывается только одна стратегия $q_l^* = G(q_k^*)$ или если любой стратегии $q_i \in G(q_k^*) \backslash q_i^*$ i-го игрока можно поставить в соответствие по крайней мере одну допустимую стратегию $\widehat{q_k} = \widehat{q_k} < q_i > \in G(q_i)$ другого игрока так, чтобы имело место соотношение*

$$J_i(\widehat{q_k} < q_i >, q_i) \leq J_i(q^*) \qquad (1)$$

Ситуацию q^* A-равновесием, если неравенства вида (1) выполняются в точке q^* для обоих $i = 1,2$.

Основным преимуществом данного понятия равновесия является то, что оно существует со сколь угодно малой точностью ε во всех задачах, удовлетворяющих допущениям 1 (см. теорему 1.2 в учебном пособии [6, с. 25]).

Таким образом удалось справиться с проблемой существования решения в конфликтных задачах.

Однако A-равновесных точек оказывается, как правило, довольно много. Чтобы сузить количество возможных решений, вводится ещё ряд конфликтных равновесий, которые постепенно сужают первоначальное множество, позволяя выбрать наиболее выгодные для всех участников точки (ситуации) на игровом множестве, устойчивые к возможным отклонениям участников.

Определение 2. *Ситуацию (точку) $q^* \in A_i$ назовём B_i-экстремальной, если образующая её стратегия другого игрока удовлетворяет условию*

$$\max_{q_k \in A_i(q_i^*)} J_k(q_i^*, q_k) = J_k(q^*), k = 1,2, k \neq i. \qquad (2)$$

Назовём ситуацию $q^* \in G$ B-равновесием, если $q^* \in B_1 \cap B_2 \stackrel{\Delta}{=} B$, где B_i - множество всех B_i-экстремальных ситуаций.

Определение 3. *Ситуацию $q^* \in B_i$ назовём $\overline{D}_i$-экстремальной, если она удовлетворяет условию*

$$\max_{q \in B_i} J_i(q) = J_i(q^*)$$

и назовём её $\overline{D}$-равновесием, если $q^* \in \overline{D_1} \cap \overline{D_2} \stackrel{\Delta}{=} \overline{D}$.

Приведённые примеры позволяют получить весьма устойчивое к отклонениям участникам решение задачи, хотя не исчерпывают всё множество предложенных в работе [6] понятий конфликтных равновесий. Однако, как уже отмечалось во введении, для практического использования крайне важно построить программную реализацию алгоритмов приближённого нахождения равновесных ситуаций.

Программная реализация методов поиска конфликтных равновесий

Для программной реализации алгоритмов нахождения A-равновесия мы воспользуемся необходимым и достаточным условием существования данного равновесия из Теоремы 1.1, сформулированной и доказанной в

работе [6, с. 25].

Теорема 1. *Для того, чтобы ситуация* $q^* \in G$ *была* A_i*- экстремальной в конфликтной задаче, удовлетворяющей допущениям 1, необходимо и достаточно удовлетворение условия*

$$J_i(q^*) \geq \sup_{q_i \in G(q_k)} \min_{q_k \in G(q_i)} J_i(q_i, q_k), i = 1,2, k \neq i. \quad (3)$$

Таким образом для того, чтобы определить, является ли некоторая точка q^* игрового множества G A_i-экстремальной необходимо и достаточно вычислить максимин (3) и сравнить с ним значение платёжной функции i-го игрока в интересующей нас точке.

В конце статьи представлен листинг с кодом функции, написанной для программной среды MATLAB, на вход которой подаются векторы возможных значений стратегий первого и второй игрока (x и y соответственно), а также матрицы значений целевых функций J_1и J_2, а в качестве результата своей работы функция выдаёт множество А-равновесных ситуаций. Аналогичным образом реализуются алгоритмы нахождения других равновесий.

Алгоритмы приближённого нахождения конфликтных равновесий в антагонистических играх на плоскости

Теперь рассмотрим использование приведённых выше алгоритмов поиска конфликтных равновесий в матричных играх для антагонистических игровых задач следующего вида.

Пусть в двумерном евклидовом пространстве Q заданы координатные оси: абсцисса q и ордината r. Также задана функция $J(q,r)$ на некотором выпуклом, ограниченном множестве G. Первый игрок выбирает стратегию (точку) из проекции $Pr_q G$ и стремится минимизировать значение функции $J(q,r)$, второй игрок выбирает стратегию r из проекции $Pr_r G$ и стремится максимизировать значение функции $J(q,r)$.

Для нахождения решения игры такого вида, построим её приближение матричной игрой. Для этого необходимо выполнить следующие шаги:

1. Получаем на вход платёжную функцию $J(q,r)$, игровое множество G, заданное системой неравенств, и требуемую точность решения ε (см. рис. 1а).
2. Находим максимальные и минимальные значения абсциссы и ординаты среди всех точек игрового множества, чтобы «вписать» его в некую прямоугольную область (см. рис. 1б).
3. Разбиваем прямоугольную область на сетку с шагом ε. Если узел входит во множество G, то считаем в нём значение функции $J(q,r)$, если не входит, то присваиваем данному узлу значение 0, и не учитываем его при определении равновесных точек. Таким образом, мы получили матрицу игры (см. рис. 1в).
4. Решаем матричную игру, используя алгоритм, приведённый в предыдущем разделе.

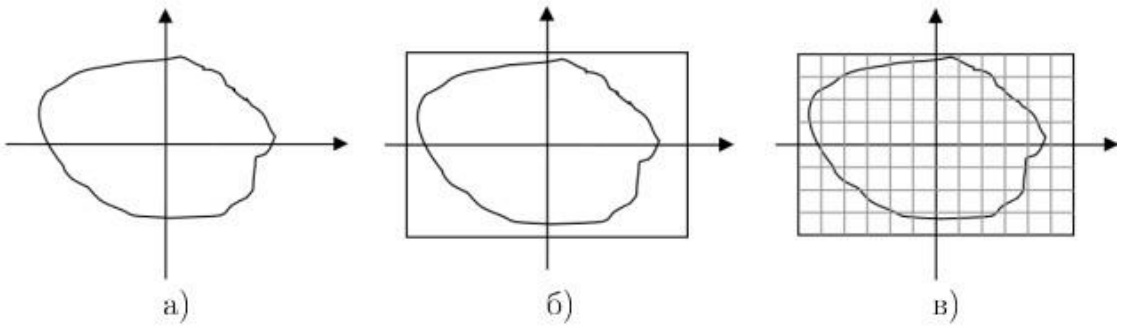

Рисунок 1 — Введение сетки с шагом ε на игровом множестве G

Использование технологии параллельных вычислений CUDA в алгоритмах поиска конфликтных равновесий

Из приведённого в листинге 1 кода видно, что в алгоритмах вычисления всех приведённых равновесий используется операции поиска максимального и минимального элемента массива. Эта операция весьма эффективно поддаётся «распараллеливанию», что позволяет существенным образом увеличить скорость выполнения алгоритма. В частности, с помощью применения технологии параллельных вычислений на графических процессорах видеокарт (GPU - *graphics processing unit*) производства компании nVidea CUDA [1] удалось решить задачу приближённого вычисления равновесий для конфликтных задач, в которых целевые функции игроков задаются непрерывными функциями, как это предполагает допущение 1.

Ниже приведены результаты работы алгоритма по нахождению A_2-равновесных ситуаций на плоскости с

платёжной функцией $J(q,r) = (q-r)^2$ при разных значениях ε. Множество G определяется следующей системой неравенств:

$$\begin{cases} q + r \le 1 \\ q - r \ge -1 \\ |q| \le 1 \\ |r| \le 1 \end{cases}$$

Точки игрового множества G отмечены серым цветом, а точки, входящие в множество A_2 – более светлым.

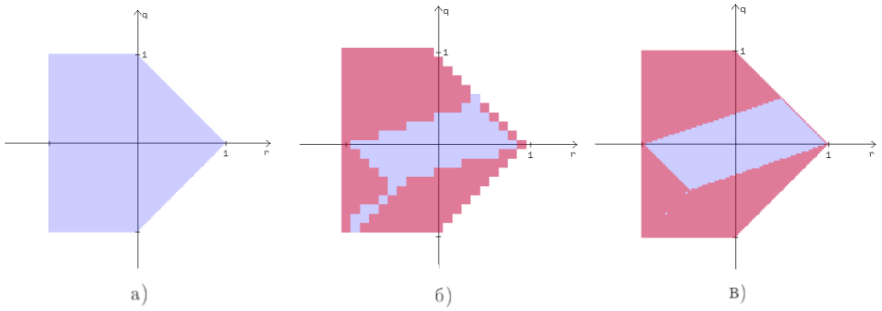

Рисунок 2. *Игровое множество G (рис. 2а) и множество A_2-равновесных точек при ε = 0.1 (рис. 2б) и ε = 0.2 (рис. 2в)*

С кодом алгоритма вычисления конфликтных равновесий для антагонистических игр на плоскости с использованием технологии параллельных вычислений на графических процессорах видеокарты компьютера CUDA можно ознакомиться по ссылке [7].

Если для решения задачи необходима точность более чем 10^{-3} ($\varepsilon \sim 0.001$), то во время построения решения приходится иметь дело с матрицами порядка 1000×1000 и более, что при выполнении программы на центральном процессоре (CPU - *central processing unit*) требует сравнительно больших затрат времени. Однако за счёт использования технологии CUDA параллельных вычислений на GPU, удаётся добиться существенного ускорения работы алгоритма.

Самым сложным с точки зрения «распараллеливания» оказался алгоритм нахождения A-равновесия, поскольку он содержит внутри себя определённое количество ветвлений, то есть таких участков кода, где разным нитям необходимо выполнять разные действия, что вызывает простаивание и снижает производительность. Тем не менее удалось получить весьма существенное ускорение работы программы (до 40 раз) по сравнению с CPU версией.

На рисунке 3 представлен график, иллюстрирующий ускорение GPU версии программы по сравнению с CPU-версией при уменьшении шага ε сетки аппроксимации непрерывной функции полезности дискретной (матричной) и, соответственно, увеличение размерности целевых матриц игроков. Все тесты проводились на видеокарте GeForce 8800 GTS (128 процессоров).

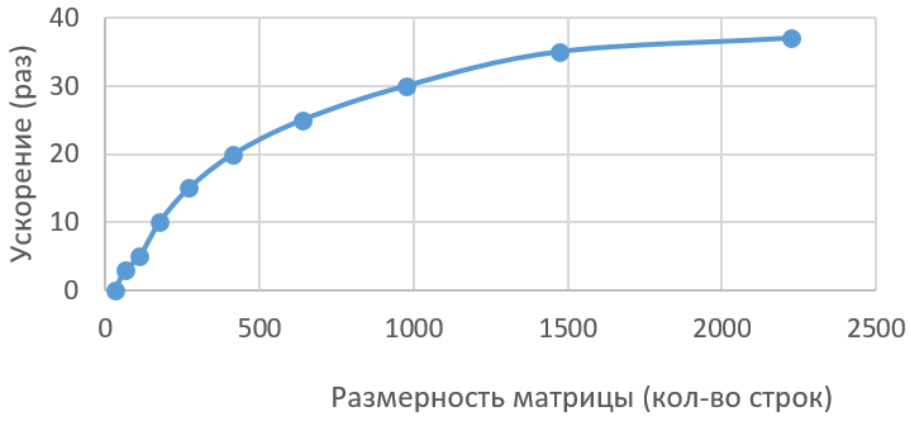

Рисунок 3. *Ускорение GPU-версии программы по сравнению с CPU-версией при увеличении размерности целевых матриц игроков*

Заключение

Таким образом удалось разработать и реализовать алгоритмы поиска решения конфликтных задач очень широкого класса, удовлетворяющих допущениям 1, что открывает широкие возможности для практического применения этих алгоритмов.

В тех случаях, когда необходимая точность вычислений требует аппроксимации непрерывных платёжных функций участниками матрицами большой размерность, а скорость выполнения алгоритма превышает допустимый временный предел, разработана версия алгоритма, оптимизированная под архитектуру CUDA параллельных вычислений на процессорах видеокарты компании nVidea.

В будущем планируется разработка алгоритмов для других конфликтных равновесий, определённых в работе [6], а также решение задач с большим количеством участников.

Листинг 1. Реализация алгоритма поиска A-равновесия в среде MATLAB

```
function [A1,A2,A]=find_A_eq(x, y, J1, J2)
%----------------------------------------
%Функция вычисления A-равновесия
%Аргументы:
%x - массив координат x (выбор первого игрока).
%y - массив координат y (выбор второго игрока).
%J1 - матрица значений целевой функции первого игрока.
%J2 - матрица значений целевой функции второго игрока.
%Возвращаемые значения:
%Матрицы A1,A2,A-равновесий размерности y строк на x столбцов
%----------------------------------------
%Определяем размер массивов координат
szx = size(x);
szy = size(y);

%Выделяем память под нулевую матрицу A1
A1 = zeros(szy, szx);

%Элемент матрицы J1 будет являться A1-равновесием тогда
%и только тогда, когда его значение будет больше или равным
%значению максимального по всем столбцам среди минимальных
%по каждому столбцу значений элементов целевой матрицы J1.
%Прежде всего определим значение искомого максимина.
%Функция min возвращает вектор минимумов всех столбцов.
si=max(min(J1));

%Цикл по всем элементам целевой матрицы первого игрока.
%Сравниваем их с найденным выше максимином.
%Элементы, которые пройдёт проверку - A1-равновесны.
for i=1:szy
for j=1:szx
if J1(i,j) >= si
A1(i,j) = 1;
end
end
end

%Выделяем память под нулевую матрицу A2
A2 = zeros(szy, szx);
%Аналогично строится матрица A2.
%Поскольку min() возвращает вектор минимумов всех столбцов,
%то прежде всего транспонируем матрицу J2.
J2T=J2';
%Находим максимин.
si=max(min(v2T));

%Цикл по всем элементам целевой матрицы второго игрока,
%Элементы, которые пройдут проверку - A2-равновесны.
```

```
for i=1:szy
for j=1:szx
if J2(i,j) >= si
A2(i,j) = 1;
end
end
end

%Находим А-равновесия.
%Прежде всего выделяем память под нулевую матрицу А
A2 = zeros(szy, szx);
%Точки, являющиеся А1 и А2-равновесиями, будут А-равновесны.
for i=1:szy
for j=1:szx
if A1(i,j) == 1
if A1(i,j) == A2(i,j)
A(i,j) = 1;
end
end
end
end

   %Визуализация полученных результатов - матриц А1, А2 и А-равновесий
   figure;
   surf(x,y, A1);
   title('A1');
   shading flat;
   figure;
   surf(x,y, A2);
   title('A2');
   shading flat;
   figure;
   surf(x,y, A);
   title('A');
   shading flat;
end
```

Список литературы

1. Боресков А. В. и др. Параллельные вычисления на GPU. Архитектура и программная модель CUDA: Учебное пособие // М.: МГУ имени М.В. Ломоносова, 2015. — 336 с.
2. Красников К.Е. Математическое моделирование некоторых социальных процессов с помощью теоретико-игровых подходов и принятие на их основе управленческих решений. Russian Technological Journal. — 2021;9(5): — 67-83
3. Красников К.Е. Программная реализация алгоритмов поиска конфликтных задач с использованием архитектуры параллельных вычислений CUDA. [Электронный ресурс]. — Режим доступа: https://github.com/KirillKrasnikov/cudaEquilibrium (дата обращения 15.01.2023).
4. Льюис Р.Д., Райфа Х. Игры и решения. М.:ИЛ. — 1961.
5. Nash J. Non-Cooperative Games // Annals of Mathematics. — 1951. V. 54. N. 2. — P. 295–320.
6. Смольяков Э.Р. Методы решения конфликтных задач. М.: Издательский отдел ф-та ВМиК МГУ им. М.В. Ломоносова. МАКС-ПРЕСС. — 2010 – 244 с.

References

1. Boreskov A. V., et all. Parallelnie vichisleniya na GPU. Arhitectura I programmnaya model CUDA: Uchebnoe posobie [Parallel computing on the GPU. CUDA Architecture and Software Model: Tutorial]. M: MSU, 2015. — 336 p.
2. Krasnikov K.E. Mathematical modeling of some social processes using game-theoretic approaches and making managerial decisions based on them. Russian Technological Journal. 2021; 9(5). P.67-83.
3. Krasnikov K.E. Software implementation of algorithms for finding conflict problems using the CUDA parallel computing architecture. Available at: https://github.com/KirillKrasnikov/cudaEquilibrium (accessed 15 January 2023).

4. Luce, R. D., Raiffa, H. Igry i resheniya [Games and decisions]. Moscow: IIL; 1961.
5. Nash J. Non-Cooperative Games // Annals of Mathematics. 1951. V. 54. N. 2. P. 295–320.
6. Smol'yakov, E.R. Metody resheniya konfliktnyh zadach: Uchebnoe posobie [Methods of solving conflict problems: Tutorial]. M.: MGU, 2010. 244 p.