**Трухин В. О. Анисич А. И. Лобанова Э. А. Нефедев К. В.**

**Ttukhin V. O. Anisich A. I. Lobanova El. A. Nefedev K. V.**

**Реализация неклассического алгоритма полного перебора модели Изинга с использованием технологии CUDA**

**Realization of non-classical algorithm of full brute-force search of Ising model using CUDA technology**

**Трухин Вячеслав Олегович** – старший преподаватель Дальневосточный федеральный университет, младший научный сотрудник, Институт прикладной математики ДВО РАН **Trukhin Viacheslav Olegovich** – senior lecturer, Far Eastern Federal University, junior researcher, Institute of Applied Mathematics, Far Eastern branch of RAS

**Анисич Александр Игоревич** – студент, Дальневосточный федеральный университет **Anisich Alexandr Igorevich** – student, Far Eastern Federal University

**Лобанова Элиза Александровна** – студент, Дальневосточный федеральный университет, лаборант, Институт прикладной математики ДВО РАН **Lobanova Eliza Alexandrovna** – student, Far Eastern Federal University, laboratory assistant, Institute of Applied Mathematics, Far Eastern branch of RAS

**Нефедев Константин Валентинович** – д.ф.-м.н., профессор, профессор департамента теоретической физики и интеллектуальных технологий ИНТиПМ ДВФУ, главный научный сотрудник ИПМ ДВО РАН. E-mail: [nefedev.kv@dvfu.ru](mailto:nefedev.kv@dvfu.ru) **Nefedev Konstantin Valentinovich** –Holder of an Advanced Doctorate (Doctor of Science) in Physico-mathematical Sciences, Full Professor, Professor at the Department of Theoretical Physics and Intelligent Technologies, Institute of High Technologies and Advanced Materials, Far Eastern Federal University, Vladivostok E-mail: [nefedev.kv@dvfu.ru](mailto:nefedev.kv@dvfu.ru)

**Аннотация.** В статье представлен авторский алгоритм полного перебора модели Изинга. Основные методы, применяемые при расчетах – это использование технологии CUDA и параллелизация на архитектуре графического процессора. Подробно описывается структура алгоритма и его применение к решению задач в статистической термодинамике. Уделено внимание сравнению скорости работы алгоритма относительно классического переборного алгоритма на Python и C++. Предложенный подход позволяет проводить точные расчеты для систем квадратной решетки спинов модели Изинга со случайным распределением обменных констант.

**Summary.** The paper presents an algorithm for complete enumeration of the Ising model. The main methods used in the calculations are the use of CUDA technology and parallelization on GPU architecture. The structure of the algorithm and its application to the solution of problems in statistical thermodynamics are described in detail. Attention is paid to the comparison of the speed of the algorithm with respect to the classical brute-force algorithm in Python and C++. The proposed approach allows to carry out accurate calculations for systems of the square lattice of spins of the Ising model with a random distribution of exchange constants.

**Ключевые слова**: статистическая физика, модель Изинга, точное решение, алгоритмы, параллельные вычисления, графический процессор.

**Keywords**: statistical physics, Ising model, exact solution, algorithms, parallel computing, GPU.

*Исследование выполнено за счет гранта Российского научного фонда № 24-71-10069,* [*https://rscf.ru/project/24-71-10069/*](https://rscf.ru/project/24-71-10069/)

*The research was supported by the Russian Science Foundation grant No. 24-71-10069,* [*https://rscf.ru/en/project/24-71-10069/*](https://rscf.ru/en/project/24-71-10069/)

УДК 537.611.2

**Введение.** Поиск точных решений для статистических моделей, с одной стороны, является критически важным для прогнозирования или описания реальных событий и экспериментов, с другой стороны – это вычислительно сложная проблема и может служить мерой эффективности как программного, так и аппаратного обеспечения.

В данной работе представлен алгоритм поиска точного решения, которое представлено в виде плотности всех возможных состояний для квадратной решетки спинов модели Изинга (МИ) со случайным распределением обменных констант [1,2], пример на рисунке 1. Эта модель не ограничивается лишь статистической физикой и может быть представлена в виде графа. Таким образом, решения данной модели вызывают интерес в различных областях науки. В настоящее время активно ведутся работы по оптимизации таких решений [3].

Термодинамические системы с фиксированным количеством частиц могут быть охарактеризованы канонической статистической суммой. Значение этой функции заключается в том, что производные от нее позволяют вычислить важные термодинамические параметры системы, такие как свободная энергия и теплоемкость. Точное решение для статистической суммы есть для двумерной МИ [4], но это решение не позволяет моделировать поведение системы со случайным распределением обменных констант или с заданным магнитным полем, но является хорошим подспорьем в проверке точности различных алгоритмов. Для решения задачи нахождения минимума в настоящее время существуют различные алгоритмы, такие как: алгоритмы машинного обучения [5], квантовый алгоритм адиабатического отжига [6], нейронные сети [7] и множество стохастических алгоритмов [8]. Однако, в отличие от упомянутых методов, стохастические алгоритмы могут тратить слишком много времени на термализацию.

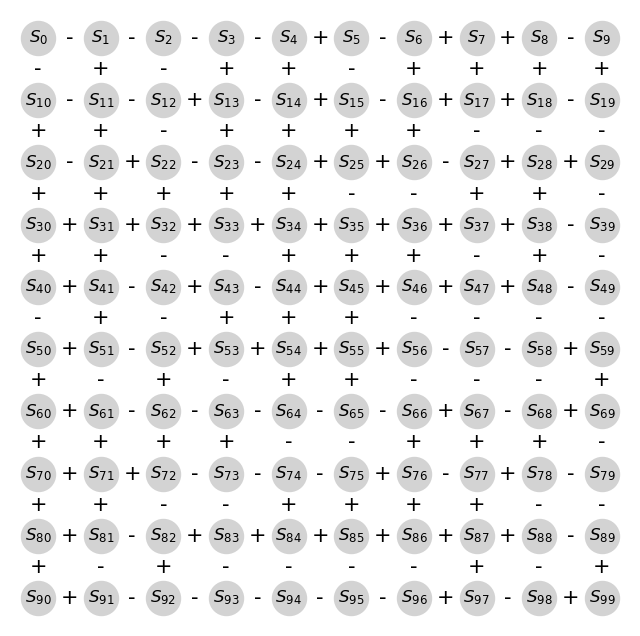


Рисунок 1. Пример решетки со случайным распределением обменных констант.

**Модель Изинга.** Простое математическое описание модели Изинга [9] вместе с отсутствием её точного решения делает её эталоном в теории сложности, так как любая NP-сложная задача может быть сведена к задаче нахождения вектора состояний для МИ со случайным распределением обменных констант [10,11,12].

В работе полная энергия взаимодействия рассчитывается по формуле

где , , – суммирование по ближайшим соседям.

**Алгоритм.** На вход алгоритма мы подаем матрицу связей J и линейный размер системы L, в результате работы мы хотим получить данные, которые опишут систему во всех ее состояниях, т.е. энергия – спиновый избыток – вырождение.

На первом этапе берутся нулевой и первый столбцы и рассчитывается их плотность состояний как независимых одномерных цепочек и для каждой заполняется 4х мерный массив: первая ось – это удвоенная максимальная энергия 1, вторая ось – удвоенный спиновый избыток 1, третья ось – это конфигурации, ее размер , и четвертая ось имеет размерность количества простых чисел, остатки от деления на которые записываются по этим координатам, размер этой оси также увеличен на 1 для проверки на совпадение. Избежать добавление этой оси можно только в случае размерности системы, не превышающей 88, в ином случае вырождение может выходить за рамки целочисленного встроенного типа данных большинства низкоуровневых языков. Устройство функции представлено в АЛГОРИТМ 1.

Второй этап заключается в соединение двух массивов состояний, полученных на

предыдущем шаге, в один новый, размерность которого будет учитывать новое количество спинов. Устройство функции представлено в АЛГОРИТМ 2.

Третий этап зацикливает первую и вторую функции до конца решетки, формулы для расчета размерности на каждом этапе весьма тривиальны. В последнем приращении у финального массива состояний можно не задавать ось конфигураций.

**АЛГОРИТМ 1**

**--------------------------------------------------------------------------**

**Функция generator\_EMC(**L, EMC, S, E\_max, M\_max, m**)**

x **:=** blockIdx.x **\*** blockDim.x **+** threadIdx.x

length\_dev **:=** 2 **^** L

E\_add **:= (**E\_max **-** 1**) /** 2

M\_add **:= (**M\_max **-** 1) **/** 2

**Для** i **от** x **до** length\_dev **с** **шагом** blockDim.x **\*** gridDim.x

E **:=** 0

M **:=** 0

**Для** j **от** 0 **до** L **-** 2

E **:=** E **-** J\_vertical**[**j **\*** L **+** m**] \*** S**[**i **\*** L **+** j**] \*** S**[**i **\*** L **+** j **+** 1**]**

M **:=** M **+** S**[**i **\*** L **+** j**]**

**КонецДля**

M **:=** M **+** S**[**i **\*** L **+** L **-** 1**]**

E **:=** E **+** E\_add

M **:=** M **+** M\_add

**atomicAdd(**EMC**[**i **\*** E\_max **\*** M\_max **\*** prime\_n\_dev **+** E **\*** M\_max **\*** prime\_n\_dev **+** M **\*** prime\_n\_dev**],** 1**)**

**Для** j **от** 1 **до** prime\_n\_dev **-** 1

**atomicAdd(**EMC**[**i **\*** E\_max **\*** M\_max **\*** prime\_n\_dev **+** E **\*** M\_max **\*** prime\_n\_dev **+** M **\*** prime\_n\_dev **+** j**],** 1**)**

**КонецДля**

**КонецДля**

**КонецФункции**

**--------------------------------------------------------------------------**

На последнем этапе мы должны раскодировать массив просуммированных остатков от деления на простые числа в массив вырождений, делаем это, основываясь на китайской теореме об остатках и расширенном алгоритме Эвклида [13].

**АЛГОРИТМ 2.**

**---------------------------------------------------------------------------------------------------**

**Функция unifying(**L, EMC\_l, E\_max\_l, M\_max\_l, EMC\_r, E\_max\_r, M\_max\_r, EMC\_created, E\_max\_created, M\_max\_created, S, column\_number**)**

x **:=** blockIdx.x **\*** blockDim.x **+** threadIdx.x

length\_dev **:=** 2 **^** L

E\_add\_l **:=** **(**E\_max\_l **-** 1**) /** 2

M\_add\_l **:=** **(**M\_max\_l **-** 1**) /** 2

E\_add\_r **:=** **(**E\_max\_r **-** 1**) /** 2

M\_add\_r **:=** **(**M\_max\_r **-** 1**) /** 2

E\_add\_created **:=** **(**E\_max\_created **-** 1**) /** 2

M\_add\_created **:=** **(**M\_max\_created **-** 1**) /** 2

**Для** conf\_l **от** x **до** length\_dev **с шагом** blockDim.x **\*** gridDim.x

**Для** E\_l **от** 0 **до** E\_max\_l **-** 1

**Для** M\_l **от** 0 **до** M\_max\_l **-** 1

**Для** conf\_r **от** 0 **до** length\_dev **-** 1

**Для** E\_r **от** 0 **до** E\_max\_r **-** 1

**Для** M\_r **от** 0 **до** M\_max\_r **-** 1

**Если** **(**EMC\_l**[**conf\_l **\*** E\_max\_l **\*** M\_max\_l **\*** prime\_n\_dev **+** E\_l **\*** M\_max\_l **\*** prime\_n\_dev **+** M\_l **\*** prime\_n\_dev**] >** 0

**И** EMC\_r[conf\_r **\*** E\_max\_r **\*** M\_max\_r **\*** prime\_n\_dev **+** E\_r **\*** M\_max\_r **\*** prime\_n\_dev **+** M\_r **\*** prime\_n\_dev**]** **>** 0**)** **Тогда**

E **:=** 0

M **:=** 0

**Для** j **от** 0 **до** L **-** 1

E **:=** E **-** J\_horizontal**[**j **\* (**L **-** 1**) +** column\_number**] \*** S**[**conf\_l **\*** L **+** j**]** **\*** S**[**conf\_r **\*** L **+** j**]**

**КонецДля**

E **:=** E **+** E\_l **-** E\_add\_l **+** E\_r **-** E\_add\_r **+** E\_add\_created

M **:=** M **+** M\_l **-** M\_add\_l **+** M\_r **-** M\_add\_r **+** M\_add\_created

**atomicAdd(&**EMC\_created**[**conf\_r **\*** E\_max\_created **\*** M\_max\_created **\***  prime\_n\_dev **+** E **\*** M\_max\_created **\*** prime\_n\_dev **+** M **\*** prime\_n\_dev**],** 1**)**

**Для** j **от** 1 **до** prime\_n\_dev **-** 1

**atomicAdd(&**EMC\_created**[**conf\_r **\*** E\_max\_created **\*** M\_max\_created **\*** prime\_n\_dev **+** E **\*** M\_max\_created **\*** prime\_n\_dev **+** M **\*** prime\_n\_dev **+** j**],** EMC\_l**[**conf\_l **\*** E\_max\_l **\*** M\_max\_l **\*** prime\_n\_dev **+**

E\_l **\*** M\_max\_l **\*** prime\_n\_dev **+** M\_l **\*** prime\_n\_dev **+** j**])**

**КонецДля**

**КонецЕсли**

**КонецДля**

**КонецДля**

**КонецДля**

**КонецДля**

**КонецДля**

**КонецДля**

**КонецФункции**

**---------------------------------------------------------------------------------------------------**

Ускорение вычислений достигается за счет расчета уникального индекса для каждого CUDA-ядра [14], что позволяет каждому потоку обрабатывать свой участок данных. Это достигается путем распараллеливания внешнего цикла, где индексы вычисляются как x blockIdx.x blockDim.xthreadIdx.x, и итерации по массиву данных с шагом blockDim.x gridDim.x, что обеспечивает эффективное использование ресурсов GPU и минимизирует время выполнения.

**Быстродействие.** В рамках анализа быстродействия алгоритма было проведено сравнение времени работы алгоритма с алгоритмами прямого перебора, написанных на Python и С++, с технологиями параллелизации по ядрам процессора таблица 1. Использование метода прямого перебора для сравнения аргументировано тем, что именно его можно взять за эталон сложности для данных методов. Таким образом, мы смогли наблюдать кратное превосходство разработанного метода, что развивает интерес дальнейших исследований в этой области.

Таблица 1. Сравнение времени, затрачиваемого на полный перебор систем спинов Изинга в зависимости от реализации алгоритма и размера системы (значения указаны в секундах).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 4 x 4 | 5 x 5 | 6 x 6 | 7 x 7 | 8 x 8 | 9 x 9 | 10 x 10 | 11 x 11 |
| Классический перебор с оптимизациями | | | | | | | | |
| C (один поток) | 0.006 | 3.9 | ― | ― | ― | ― | ― | ― |
| С (+ OpenMP) | 0.008 | 0.28 | 499 | ― | ― | ― | ― | ― |
| Glaurung алгоритм | | | | | | | | |
| Python (один поток) | 4.09 | ― | ― | ― | ― | ― | ― | ― |
| Python (+numba) | 0.046 | 66 | ― | ― | ― | ― | ― | ― |
| C (один поток) | 0.076 | 0.5 | 5.5 | 72.84 | 756 | 7242 | ― | ― |
| С (+ OpenMP) | 0.074 | 0.21 | 0.61 | 3.3 | 23 | 206 | 2085 |  |
| CUDA |  |  |  |  |  |  |  |  |

**Заключение.** В результате работы алгоритма мы получаем уникальные данные, характеризующие исследуемую систему. Уникальность заключается в том, что можно определить, глобальный ли минимум был найден с помощью алгоритмов, не являющихся точными, например, Монте-Карло или генетического алгоритма [15]. Также это позволяет определить оптимальное количество шагов для метода Метрополиса, необходимых для нахождения глобального минимума.

Полученные с помощью алгоритма характеристики состояний позволяют рассчитать свойства спиновых систем, которые без полной плотности состояний получить нельзя. В том числе удалось классифицировать решетку Эдвардса-Андерсона на ферромагнитное, антиферромагнитное состояния и состояние спинового стекла. А также провести численный эксперимент по определению фазовых свойств таких решеток в поле.

1. Roma F and Risau-Gusman, S and Ramirez-Pastor, AJ and Nieto, F and Vogel, EE Physical Review B—Condensed Matter and Materials Physics Ground-state topology of the Edwards-Anderson J spin glass model 82 21 214-401 2010
2. Katzgraber, Helmut G and Lee, Lek Wee, Correlation length of the two dimensional Ising spin glass with bimodal interactions, Physical Revive B – condensed Matter and Materials Physics, 71:13, (2005), 134-404
3. Romero, Joshua and Bisson, Mauro and Fatica, Massimiliano and Bernaschi, Massimo Computer Physics Communicationspaper High performance implementations of the 2D Ising model on GPUs (2020) 256
4. Onsager, Lars. (1944). Crystal statistics. I. A two-dimensional model with an order-disorder transition. Physical Review, 65(3-4), 117. APS.
5. Maren, Alianna J. (1991). A logical topology of neural networks. In Proceedings of the Second Workshop on Neural Networks.
6. Grant, Erica K, Humble, Travis S. (2020). Adiabatic quantum computing and quantum annealing. In Oxford Research Encyclopedia of Physics
7. Korol, Alyona Olegovna, Captain, Vitaly Yurievich. (2021). Neural network for determining the Curie temperature of the two-dimensional Ising model. Far Eastern Mathematical Journal, 21(1), 51-60. Institute of Applied Mathematics, Far East Branch of RAS.
8. Janke, Wolfhard. (2008). Monte Carlo methods in classical statistical physics. In Computational many-particle physics (pp. 79-140). Springer.
9. E. Ising, Z. Phys. XXXI (1925)
10. Markovich, LA. (2019). Parallel algorithm based on the Ising model for solving combinatorial optimization problems. In Information Technology and Systems 2019 (pp. 350-358).
11. C. H. Papadimitriou, The Euclidean travelling salesman problem is np-complete, Theoretical computer sciences 4 (3) (1977)
12. Karp, Richard M. (2010). Reducibility among combinatorial problems. Springer.
13. Katz, Victor J. (2007). The Mathematics of Egypt, Mesopotamia, China, India, and Islam: A Sourcebook. Princeton University Press.
14. NVIDIA CUDA, <HTTPS://DEVELOPER.NVIDIA.COM/CUDA-ZONE>.
15. Panchenko, Tatyana Vyacheslavovna, Tarasevich, Yuri Yurievich. (2007). Comparative analysis of the efficiency of application of genetic algorithms and Metropolis algorithm in problems of solid state physics. Computational Methods and Programming, 8, 77-87.