

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Физико-технологический институт

Кафедра «Технической физики»

Оценка

Преподаватель

Кашин И.В.

**ЧИСЛЕННЫЙ РАСЧЕТ ГОЛОСОВАНИЯ ДЕПУТАТОВ С МНОЖЕСТВЕННЫМ ВЫБОРОМ**

ОТЧЕТ

по лабораторной работе №4

|  |  |  |
| --- | --- | --- |
| Студент | Черняков Матвей Сергеевич | ФИО студента |

|  |
| --- |
| Специальность (направление подготовки) |
| 09.03.02 Информационные системы и технологии | |

|  |  |
| --- | --- |
| Группа | Фт-420008 |

Екатеринбург

2025

**ОГЛАВЛЕНИЕ**

[ОПИСАНИЕ ЗАДАЧИ 3](#_Toc210667119)

[ПРИНЦИП ПАРАЛЛЕЛИЗАЦИИ 5](#_Toc210667120)

[РЕШЕНИЕ ЗАДАЧИ 6](#_Toc210667121)

[ОЦЕНКА ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛИЗАЦИИ 8](#_Toc210667122)

[ПРИЛОЖЕНИЕ А – ЛИСТИНГ КОДА 10](#_Toc210667123)

[ПРИЛОЖЕНИЕ Б – ЭКСПЕРИМЕНТ 1 16](#_Toc210667129)

[ПРИЛОЖЕНИЕ В – ЭКСПЕРИМЕНТ 2 18](#_Toc210667130)

ОПИСАНИЕ ЗАДАЧИ

В задаче рассматривается поиск минимальных конфигураций голосования депутатов. Целью является сравнение точности и производительности двух способов вычислений — однопоточного и многопоточного. Целевая функция описывается следующим образом:

где – воля депутата (-2 или 2), – голос депутата (+2 агрессивный, +1 умеренно агрессивный, 0 нейтральный, -1 умеренно консервативный, -2 консервативный), – значение из матрицы взаимоотношений депутатов (случайное значение взаимоотношения находится в интервале [-2;2]), , – депутаты для которых берется попарное взаимоотношение.

Для данного количества депутатов N получается следующее количество всевозможных вариантов значений N:

Для того, чтобы посчитать локальные минимумы для функций по Е выбирается случайная конфигурация голосов депутатов и изменяется случайный голос одного из депутатов на +1 или -1. Далее мы вычисляем значения E в этих двух точках и обозначим следующие значения:

где – начальная конфигурация, – различие в голосе депутата на +1/-1.

Обозначим разницу за .

Тогда:

При (1) произведем переход в пробную точку (меняем на новую конфигурацию).

При (2) переход, но с вероятностью p:

Где T – мера хаоса. T = 200, для которой каждую 1000-ю операцию до .

Для розыгрыша такой ситуации зададим число R такое, что:

При (3) производим переход.

При (4) не производим переход, конец итерации.

ПРИНЦИП ПАРАЛЛЕЛИЗАЦИИ

Задача разбивается на части, которые распределяются между несколькими процессами. Каждый процесс выполняет свой участок работы, а затем результаты собираются и объединяются в итоговый ответ.

В рамках поставленной задачи это сводится к разделению количества вычисляемых значений E на равные части, где каждый поток забирает часть вариантов голосов депутатов и вычисляет E итерационно.

РЕШЕНИЕ ЗАДАЧИ

В коде реализуются две основные функции воркеров и дополнительные функции.

Функция symmetric\_random\_matrix(n, seed) – создает массив попарных взаимоотношений депутатов.

def symmetric\_random\_matrix(n, seed=None) -> np.ndarray:  
 *"""  
 Возвращает n x n симметричную матрицу:  
 - диагональ = 0  
 - элементы a[i,j] = a[j,i] = случайный float в [-2, 2], округлённый до 12 знаков  
 Параметр seed для воспроизводимости.  
 """* if seed is not None:  
 rng = np.random.default\_rng(seed)  
 else:  
 rng = np.random.default\_rng()  
 *# верхний треугольник без диагонали* upper = rng.uniform(-2.0, 2.0, size=(n, n))  
 *# обнулим нижний треугольник и диагональ (сохраняем только strict upper)* upper = np.triu(upper, k=1)  
 *# зеркалим в нижний треугольник* mat = upper + upper.T  
 *# диагональ нули (на всякий случай)* np.fill\_diagonal(mat, 0.0)  
 *# округление до 12 знаков после запятой* mat = np.round(mat, 12)  
 return mat

Функция generate\_deputies\_will – создает случайные предпочтения депутатов.

def generate\_deputies\_will(deputies) -> np.ndarray:  
 return np.random.choice([-2, 2, 0], size=deputies)

Функция generate\_deputies\_votes – создает случайные голоса депутатов.

def generate\_deputies\_votes(deputies) -> np.ndarray:  
 return np.random.choice([-2, -1, 0, 1, 2], size=deputies)

Функция calculate\_E – вычисляет значение E.

def calculate\_E(N, A, B, V):  
 E = np.sum(A \* V)  
 for n in range(N):  
 for m in range(n + 1, N):  
 E += B[n, m] \* V[n] \* V[m]  
 return E

Далее для экспериментов используется вызов этих функций и функций воркеров с помощью pool из библиотеки multiprocessing.

for workers in range(1, workers + 1):  
 base\_count = total\_tasks // workers  
 remainder = total\_tasks % workers  
 counts = [base\_count + 1 if i < remainder else base\_count for i in range(workers)]  
 all\_tasks = [(N, A, B, count, T) for count in counts]  
 total\_elapsed, min\_E = np.inf, np.inf  
  
 with multiprocessing.Pool(processes=workers) as pool:  
 results = pool.starmap(worker, all\_tasks)

ОЦЕНКА ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛИЗАЦИИ

Ключевыми параметрами является количество ядер (12) и количество потоков (24). Для данных характеристик было проведено два тестирования:

1. Сравнение идеального и условного времени вычисления 10 минимумов функций при M от 1 до 24. Результаты тестирования представлены на графике (рисунок 1). Точные значения представлены в приложении Б;
2. Сравнение идеального и условного времени поиска равного или минимального Е из эксперимента 1 при M от 1 до 24. Результаты тестирования представлены на графике (рисунок 2). Точные значения представлены в приложении В.

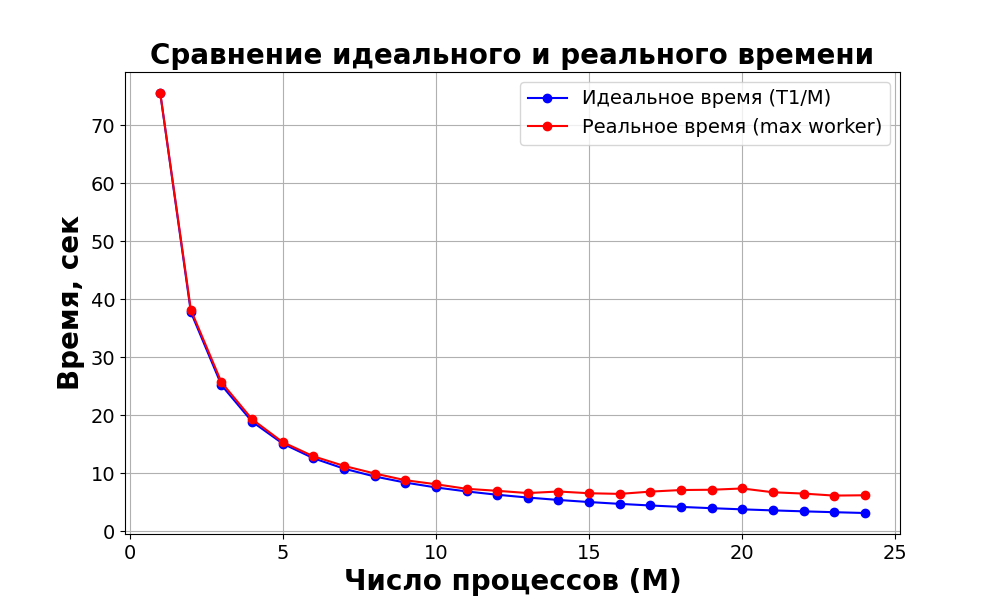


Рисунок 1 – График эксперимента 1

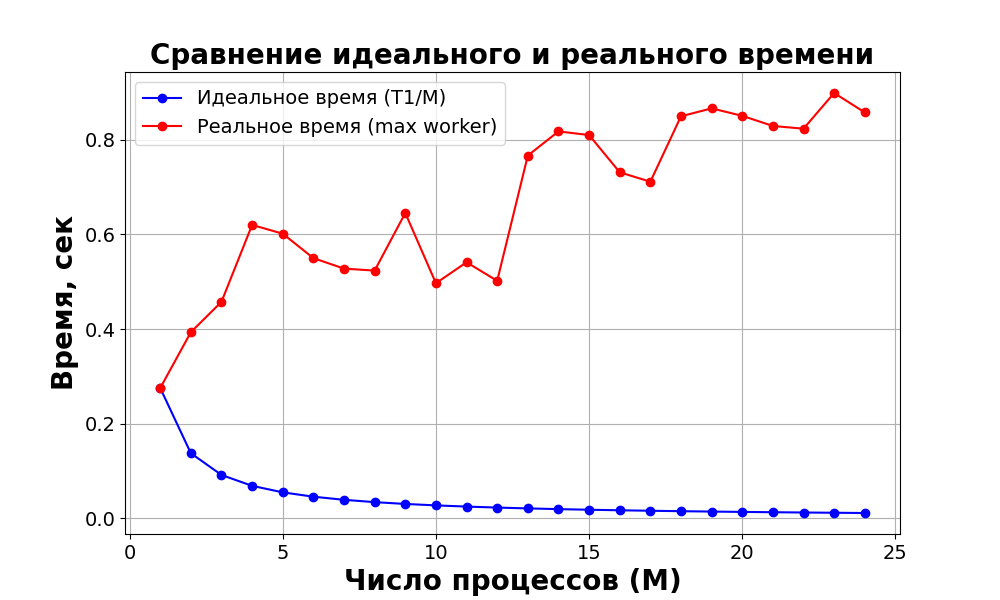


Рисунок 2 – Графики эксперимента 2

По графику для эксперимента 1 видно, что реальное время достаточно близко к идеальному и отклоняется на большом количестве потоков меньше чем на 1 секунду.

ПРИЛОЖЕНИЕ А – ЛИСТИНГ КОДА

import multiprocessing  
import time  
from collections import Counter  
from operator import itemgetter  
  
import matplotlib.pyplot as plt  
import numpy as np  
  
  
def generate\_deputies\_will(deputies) -> np.ndarray:  
 return np.random.choice([-2, 2, 0], size=deputies)  
  
  
def generate\_deputies\_votes(deputies) -> np.ndarray:  
 return np.random.choice([-2, -1, 0, 1, 2], size=deputies)  
  
  
def symmetric\_random\_matrix(n, seed=None) -> np.ndarray:  
 *"""  
 Возвращает n x n симметричную матрицу:  
 - диагональ = 0  
 - элементы a[i,j] = a[j,i] = случайный float в [-2, 2], округлённый до 12 знаков  
 Параметр seed для воспроизводимости.  
 """* if seed is not None:  
 rng = np.random.default\_rng(seed)  
 else:  
 rng = np.random.default\_rng()  
 *# верхний треугольник без диагонали* upper = rng.uniform(-2.0, 2.0, size=(n, n))  
 *# обнулим нижний треугольник и диагональ (сохраняем только strict upper)* upper = np.triu(upper, k=1)  
 *# зеркалим в нижний треугольник* mat = upper + upper.T  
 *# диагональ нули (на всякий случай)* np.fill\_diagonal(mat, 0.0)  
 *# округление до 12 знаков после запятой* mat = np.round(mat, 12)  
 return mat  
  
  
def calculate\_E(N, A, B, V):  
 E = np.sum(A \* V)  
 for n in range(N):  
 for m in range(n + 1, N):  
 E += B[n, m] \* V[n] \* V[m]  
 return E  
  
  
def worker(N, A, B, count, T):  
 *"""Задача №1"""* min\_E = np.inf  
 V = []  
 transitions = {  
 2: [1],

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А – ЛИСТИНГ КОДА

-2: [-1],  
 1: [2, 0],  
 -1: [-2, 0],  
 0: [1, -1],  
 }  
 t0 = time.perf\_counter()  
 for i in range(count):  
 V = generate\_deputies\_votes(N)  
 V0 = V.copy()  
 T\_local = T  
 iteration = 1  
 E = 0  
 while T\_local > 0.01:  
 E = calculate\_E(N, A, B, V)  
 random\_deputy = np.random.randint(0, N)  
 current = V[random\_deputy]  
 possible\_moves = transitions[current]  
  
 if len(possible\_moves) == 2:  
 best\_V = V.copy()  
 best\_E = E  
 for new\_val in possible\_moves:  
 V\_test = best\_V.copy()  
 V\_test[random\_deputy] = new\_val  
 E\_new = calculate\_E(N, A, B, V\_test)  
 if E\_new < best\_E:  
 best\_E = E\_new  
 best\_V = V\_test  
 else:  
 V\_test = V.copy()  
 V\_test[random\_deputy] = possible\_moves[0]  
 E\_new = calculate\_E(N, A, B, V\_test)  
 best\_E = E\_new  
 best\_V = V\_test  
  
 delta\_E = best\_E - E  
 if delta\_E < 0:  
 V = best\_V  
 else:  
 p = np.exp(-delta\_E / T\_local)  
 R = np.random.random() *# (0, 1)* if p >= R:  
 V = best\_V  
 else:  
 V = V0.copy()  
 iteration += 1  
 if iteration == 10: *# 1000* iteration = 1  
 T\_local = T\_local \* 0.2 *# 0.99  
 # print(f"Высчитано {i + 1}-ое значение E = {E}")* min\_E = min(min\_E, E)  
  
 t1 = time.perf\_counter()  
 elapsed = t1 - t0  
 return [min\_E, elapsed, V]

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А – ЛИСТИНГ КОДА

def worker\_task2(N, A, B, T, E\_target, stop\_event):  
 *"""  
 Задача №2: поиск конфигурации с E <= E\_target.  
 Цикл продолжается, пока не найдено E <= E\_target.  
 T\_local не уменьшается ниже 0.01  
 """* min\_E = np.inf  
 elapsed = 0  
 V = 0  
 transitions = {  
 2: [1],  
 -2: [-1],  
 1: [2, 0],  
 -1: [-2, 0],  
 0: [1, -1],  
 }  
  
 t0 = time.perf\_counter()  
  
 while min\_E > E\_target and not stop\_event.is\_set():  
 V = generate\_deputies\_votes(N)  
 V0 = V.copy()  
 T\_local = T  
 iteration = 1  
  
 while True:  
 E = calculate\_E(N, A, B, V)  
 if E <= E\_target:  
 min\_E = min(min\_E, E)  
 stop\_event.set()  
 break  
  
 random\_deputy = np.random.randint(0, N)  
 current = V[random\_deputy]  
 possible\_moves = transitions[current]  
  
 best\_V = V.copy()  
 best\_E = E  
  
 for new\_val in possible\_moves:  
 V\_test = V.copy()  
 V\_test[random\_deputy] = new\_val  
 E\_new = calculate\_E(N, A, B, V\_test)  
 if E\_new < best\_E:  
 best\_E = E\_new  
 best\_V = V\_test  
  
 delta\_E = best\_E - E  
 if delta\_E < 0:  
 V = best\_V  
 else:  
 p = np.exp(-delta\_E / T\_local)  
 R = np.random.random() *# (0, 1)* if p >= R:  
 V = best\_V  
 else:  
 V = V0.copy()  
 iteration += 1

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А – ЛИСТИНГ КОДА

if iteration == 10: *# 1000* iteration = 1  
 T\_local = T\_local \* 0.2 *# 0.99* T\_local = max(T\_local, 0.01)  
  
 t1 = time.perf\_counter()  
 elapsed = t1 - t0  
 return [min\_E, elapsed, V]  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 N = 100  
  
 *# Фиксированные коэффициенты* A = generate\_deputies\_will(N) *# Воля депутата* B = symmetric\_random\_matrix(N) *# Взаимоотношения депутатов  
  
 # ---------------- PARALLEL TASK №1 ----------------* workers = multiprocessing.cpu\_count() *# 24* chunk\_size = 10  
 total\_tasks = workers \* chunk\_size  
 T = 200  
  
 real\_times = []  
 ideal\_times = []  
 result\_min\_E = np.inf  
  
 T1 = 0  
  
 print("Параллельная задача №1...")  
 for workers in range(1, workers + 1):  
 base\_count = total\_tasks // workers  
 remainder = total\_tasks % workers  
 counts = [base\_count + 1 if i < remainder else base\_count for i in range(workers)]  
 all\_tasks = [(N, A, B, count, T) for count in counts]  
 total\_elapsed, min\_E = np.inf, np.inf  
  
 with multiprocessing.Pool(processes=workers) as pool:  
 results = pool.starmap(worker, all\_tasks)  
  
 conf\_res = []  
  
 for E, elapsed, configurations in results:  
 if E < min\_E:  
 conf\_res = configurations  
 min\_E = min(min\_E, E)  
 if elapsed != 0:  
 total\_elapsed = min(total\_elapsed, elapsed) *# min != 0!!!!!!!!* real\_times.append(total\_elapsed)  
 if workers == 1:  
 T1 = total\_elapsed  
 ideal\_times.append(T1 / workers)  
 result\_min\_E = min(result\_min\_E, min\_E)

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А – ЛИСТИНГ КОДА

print(f"Потоков: {workers:2d} | chunk\_size: {base\_count:3d} | "  
 f"Время: {total\_elapsed:.4f} сек | E\_min: {min\_E:.4f} | "  
 f"Топ голосов: {sorted(Counter(conf\_res).most\_common(5), key=itemgetter(1))}")  
 print(f"Минимальное E - {result\_min\_E:.4f}")  
  
 *# ---------------- ГРАФИК №1 ----------------* M\_values = list(range(1, workers + 1))  
 fig, ax = plt.subplots(figsize=(10, 6))  
 ax.plot(M\_values, ideal\_times, "bo-", label="Идеальное время (T1/M)")  
 ax.plot(M\_values, real\_times, "ro-", label="Реальное время (max worker)")  
  
 ax.set\_title("Сравнение идеального и реального времени", fontsize=20, fontweight='bold')  
 ax.set\_xlabel("Число процессов (M)", fontsize=20, fontweight='bold')  
 ax.set\_ylabel("Время, сек", fontsize=20, fontweight='bold')  
  
 ax.tick\_params(axis='both', which='major', labelsize=14)  
 ax.legend(fontsize=14)  
 ax.grid(True)  
  
 plt.show()  
  
 *# ---------------- PARALLEL TASK №2 ----------------* real\_times2 = []  
 ideal\_times2 = []  
 total\_elapsed2 = 0  
 result\_min\_E2 = np.inf  
 print("Параллельная задача №2...")  
 for workers in range(1, workers + 1):  
 stop\_event = multiprocessing.Manager().Event()  
 all\_tasks2 = [(N, A, B, T, result\_min\_E, stop\_event) for \_ in range(workers)]  
  
 with multiprocessing.Pool(processes=workers) as pool:  
 results2 = pool.starmap(worker\_task2, all\_tasks2)  
  
 conf\_res2 = []  
  
 total\_elapsed2 = 0  
 min\_E2 = np.inf  
 for E, elapsed, configurations in results2:  
 if E < min\_E2:  
 conf\_res2 = configurations  
 min\_E2 = min(min\_E2, E)  
 total\_elapsed2 = max(total\_elapsed2, elapsed)  
  
 real\_times2.append(total\_elapsed2)  
 if workers == 1:  
 T1 = total\_elapsed2  
 ideal\_times2.append(T1 / workers)  
 result\_min\_E2 = min(result\_min\_E2, min\_E2)

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А – ЛИСТИНГ КОДА

print(f"Потоков: {workers:2d} | Время: {total\_elapsed2:.4f} сек | "  
 f"E\_min2: {min\_E2:.4f} | Топ голосов: {sorted(Counter(conf\_res2).most\_common(5), key=itemgetter(1))}")  
 print(f"Минимальное E2 - {result\_min\_E2:.4f}")  
  
 *# ---------------- ГРАФИК №2 ----------------* fig, ax = plt.subplots(figsize=(10, 6))  
 ax.plot(M\_values, ideal\_times2, "bo-", label="Идеальное время (T1/M)")  
 ax.plot(M\_values, real\_times2, "ro-", label="Реальное время (max worker)")  
  
 ax.set\_title("Сравнение идеального и реального времени", fontsize=20, fontweight='bold')  
 ax.set\_xlabel("Число процессов (M)", fontsize=20, fontweight='bold')  
 ax.set\_ylabel("Время, сек", fontsize=20, fontweight='bold')  
  
 ax.tick\_params(axis='both', which='major', labelsize=14)  
 ax.legend(fontsize=14)  
 ax.grid(True)  
  
 plt.show()

ПРИЛОЖЕНИЕ Б – ЭКСПЕРИМЕНТ 1

Параллельная задача №1...

Потоков: 1 | chunk\_size: 240 | Время: 75.5498 сек | E\_min: -488.3529 | Топ голосов: [(1, 13), (-1, 16), (2, 18), (-2, 24), (0, 29)]

Потоков: 2 | chunk\_size: 120 | Время: 38.1715 сек | E\_min: -429.2147 | Топ голосов: [(-1, 14), (1, 16), (0, 20), (2, 24), (-2, 26)]

Потоков: 3 | chunk\_size: 80 | Время: 25.6590 сек | E\_min: -479.9051 | Топ голосов: [(-2, 15), (0, 15), (2, 22), (-1, 23), (1, 25)]

Потоков: 4 | chunk\_size: 60 | Время: 19.3014 сек | E\_min: -491.3596 | Топ голосов: [(2, 17), (-1, 18), (1, 20), (-2, 22), (0, 23)]

Потоков: 5 | chunk\_size: 48 | Время: 15.3562 сек | E\_min: -510.2180 | Топ голосов: [(-2, 17), (-1, 17), (1, 18), (0, 19), (2, 29)]

Потоков: 6 | chunk\_size: 40 | Время: 12.9244 сек | E\_min: -521.2914 | Топ голосов: [(1, 19), (2, 19), (0, 20), (-1, 21), (-2, 21)]

Потоков: 7 | chunk\_size: 34 | Время: 11.2650 сек | E\_min: -484.9769 | Топ голосов: [(0, 15), (1, 18), (-1, 19), (-2, 23), (2, 25)]

Потоков: 8 | chunk\_size: 30 | Время: 9.9564 сек | E\_min: -493.1612 | Топ голосов: [(-2, 17), (0, 19), (1, 20), (-1, 21), (2, 23)]

Потоков: 9 | chunk\_size: 26 | Время: 8.8011 сек | E\_min: -585.0382 | Топ голосов: [(0, 15), (2, 16), (-2, 18), (-1, 24), (1, 27)]

Потоков: 10 | chunk\_size: 24 | Время: 8.1056 сек | E\_min: -607.4283 | Топ голосов: [(2, 14), (-2, 17), (-1, 18), (1, 23), (0, 28)]

Потоков: 11 | chunk\_size: 21 | Время: 7.3180 сек | E\_min: -417.6615 | Топ голосов: [(0, 16), (2, 16), (1, 21), (-1, 22), (-2, 25)]

Потоков: 12 | chunk\_size: 20 | Время: 6.9734 сек | E\_min: -525.4604 | Топ голосов: [(1, 16), (-2, 18), (2, 21), (0, 22), (-1, 23)]

Потоков: 13 | chunk\_size: 18 | Время: 6.5831 сек | E\_min: -519.2185 | Топ голосов: [(1, 11), (-1, 17), (2, 19), (0, 25), (-2, 28)]

Потоков: 14 | chunk\_size: 17 | Время: 6.8533 сек | E\_min: -504.9714 | Топ голосов: [(1, 9), (-1, 20), (0, 22), (-2, 22), (2, 27)]

Потоков: 15 | chunk\_size: 16 | Время: 6.5551 сек | E\_min: -522.2768 | Топ голосов: [(2, 12), (-1, 21), (0, 21), (1, 23), (-2, 23)]

Потоков: 16 | chunk\_size: 15 | Время: 6.4416 сек | E\_min: -604.1470 | Топ голосов: [(-2, 16), (0, 16), (2, 21), (1, 21), (-1, 26)]

Потоков: 17 | chunk\_size: 14 | Время: 6.8342 сек | E\_min: -413.3398 | Топ голосов: [(2, 14), (-2, 16), (0, 22), (-1, 23), (1, 25)]

Потоков: 18 | chunk\_size: 13 | Время: 7.1078 сек | E\_min: -629.9213 | Топ голосов: [(-2, 15), (0, 18), (-1, 20), (2, 20), (1, 27)]

Потоков: 19 | chunk\_size: 12 | Время: 7.1557 сек | E\_min: -670.8433 | Топ голосов: [(0, 11), (-2, 20), (2, 22), (-1, 23), (1, 24)]

Потоков: 20 | chunk\_size: 12 | Время: 7.3766 сек | E\_min: -504.4005 | Топ голосов: [(-1, 13), (-2, 18), (2, 20), (1, 23), (0, 26)]

Потоков: 21 | chunk\_size: 11 | Время: 6.7227 сек | E\_min: -527.1244 | Топ голосов: [(-1, 12), (0, 20), (2, 21), (-2, 23), (1, 24)]

Потоков: 22 | chunk\_size: 10 | Время: 6.4946 сек | E\_min: -556.9645 | Топ голосов: [(2, 15), (1, 19), (0, 19), (-1, 23), (-2, 24)]

Потоков: 23 | chunk\_size: 10 | Время: 6.1479 сек | E\_min: -528.4742 | Топ голосов: [(-1, 16), (-2, 17), (0, 19), (2, 20), (1, 28)]

Потоков: 24 | chunk\_size: 10 | Время: 6.2056 сек | E\_min: -462.5466 | Топ голосов: [(2, 17), (-2, 18), (-1, 21), (1, 22), (0, 22)]

Минимальное E - -670.8433

ПРИЛОЖЕНИЕ В – ЭКСПЕРИМЕНТ 2

Параллельная задача №2...

Потоков: 1 | Время: 0.2755 сек | E\_min2: -686.5933 | Топ голосов: [(0, 16), (-1, 18), (1, 19), (-2, 21), (2, 26)]

Потоков: 2 | Время: 0.3935 сек | E\_min2: -677.3379 | Топ голосов: [(-2, 16), (2, 18), (-1, 20), (1, 21), (0, 25)]

Потоков: 3 | Время: 0.4577 сек | E\_min2: -681.4659 | Топ голосов: [(-1, 13), (0, 18), (1, 22), (2, 22), (-2, 25)]

Потоков: 4 | Время: 0.6201 сек | E\_min2: -678.4243 | Топ голосов: [(0, 9), (-1, 13), (-2, 22), (2, 27), (1, 29)]

Потоков: 5 | Время: 0.6015 сек | E\_min2: -687.8856 | Топ голосов: [(1, 16), (2, 17), (0, 18), (-1, 19), (-2, 30)]

Потоков: 6 | Время: 0.5499 сек | E\_min2: -699.3986 | Топ голосов: [(0, 16), (1, 17), (-1, 18), (-2, 22), (2, 27)]

Потоков: 7 | Время: 0.5278 сек | E\_min2: -704.4299 | Топ голосов: [(1, 13), (0, 19), (2, 21), (-1, 23), (-2, 24)]

Потоков: 8 | Время: 0.5236 сек | E\_min2: -683.4988 | Топ голосов: [(0, 10), (-1, 16), (2, 20), (-2, 24), (1, 30)]

Потоков: 9 | Время: 0.6453 сек | E\_min2: -690.6320 | Топ голосов: [(1, 12), (0, 15), (2, 19), (-1, 22), (-2, 32)]

Потоков: 10 | Время: 0.4969 сек | E\_min2: -695.1545 | Топ голосов: [(-1, 12), (1, 18), (2, 19), (0, 20), (-2, 31)]

Потоков: 11 | Время: 0.5412 сек | E\_min2: -713.2478 | Топ голосов: [(1, 13), (0, 16), (-2, 22), (-1, 23), (2, 26)]

Потоков: 12 | Время: 0.5020 сек | E\_min2: -719.2299 | Топ голосов: [(-1, 13), (1, 16), (0, 22), (-2, 22), (2, 27)]

Потоков: 13 | Время: 0.7666 сек | E\_min2: -703.1555 | Топ голосов: [(1, 15), (0, 16), (-1, 21), (2, 24), (-2, 24)]

Потоков: 14 | Время: 0.8176 сек | E\_min2: -694.5485 | Топ голосов: [(2, 14), (-1, 19), (0, 19), (1, 23), (-2, 25)]

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В

Потоков: 15 | Время: 0.8100 сек | E\_min2: -711.4894 | Топ голосов: [(2, 16), (0, 18), (-1, 19), (1, 23), (-2, 24)]

Потоков: 16 | Время: 0.7313 сек | E\_min2: -698.3504 | Топ голосов: [(1, 17), (2, 19), (0, 20), (-1, 21), (-2, 23)]

Потоков: 17 | Время: 0.7114 сек | E\_min2: -713.9877 | Топ голосов: [(0, 15), (1, 19), (-1, 21), (2, 22), (-2, 23)]

Потоков: 18 | Время: 0.8496 сек | E\_min2: -712.6904 | Топ голосов: [(2, 15), (0, 18), (1, 19), (-1, 22), (-2, 26)]

Потоков: 19 | Время: 0.8663 сек | E\_min2: -688.6048 | Топ голосов: [(1, 11), (-1, 17), (2, 20), (0, 24), (-2, 28)]

Потоков: 20 | Время: 0.8505 сек | E\_min2: -709.0121 | Топ голосов: [(1, 16), (0, 20), (-2, 21), (2, 21), (-1, 22)]

Потоков: 21 | Время: 0.8292 сек | E\_min2: -698.0789 | Топ голосов: [(2, 14), (-1, 17), (1, 18), (-2, 25), (0, 26)]

Потоков: 22 | Время: 0.8234 сек | E\_min2: -694.5799 | Топ голосов: [(-1, 14), (0, 16), (1, 16), (2, 27), (-2, 27)]

Потоков: 23 | Время: 0.8988 сек | E\_min2: -697.3615 | Топ голосов: [(1, 9), (0, 13), (-1, 23), (2, 26), (-2, 29)]

Потоков: 24 | Время: 0.8578 сек | E\_min2: -709.8359 | Топ голосов: [(1, 11), (-1, 19), (0, 21), (2, 21), (-2, 28)]

Минимальное E2 - -719.2299