## МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЕ РФ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

## “ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Лабораторная работа №2

По дисциплине: «Эвристические методы и алгоритмы»

Выполнил:

Студент группы ВПР31

Андросов И.А.

Проверил:

Проф. Кобак В.Г.

Ростов-на-Дону 2023 г.

# Введение

Задачи проектирования и управления в системах, для которых необходимо распределение работы между параллельно работающими разнородными вычислительными устройствами занимают значимое место в теории построения расписаний. Практическая актуальность таких задач определяется существенными возможностями экономии машинного времени и вытекающими функциональными и эксплуатационными преимуществами.

Теоретическая сложность нахождения наилучшего распределения связана с необходимостью решения экстремальных задач комбинаторного типа, требующих больших вычислительных ресурсов, так что эффект от нахождения близкого к оптимальному, с точки зрения времени выполнения, распределения может быть сведен на нет затратами на его получение.

В настоящем руководстве приводятся методы получения расписаний, приводящие к небольшим затратам на вычисление за счет отказа от получения оптимального решения, но в тоже время позволяющие найти приемлемое решение, близкое к оптимальному.

# Постановка задачи

Имеется  независимых работ , которые необходимо распределить на  параллельно работающих разнородных устройств  по критерию , где - время завершения работы процессора . Каждое устройство  выполняет только одну работу в определенный момент времени и выполнение задания не прерывается для передачи на другой процессор. Известно (вес) время выполнения  задания  на любом из устройств . Требуется найти такое распределение заданий по процессорам, при котором суммарное время выполнения заданий на каждом из процессоров было бы минимальным.

Получение оптимального распределения в такой постановке приводит к громоздким вычислениям, требующим значительного времени машинного счета, поэтому цель – продемонстрировать алгоритмы, с помощью которого можно находить с малыми затратами достаточно приемлемое решение.

# Результаты тестирования программы

СОРТИРОВКА ПО УБЫВАНИЮ

Matrix:

| Field 1 | Field 2 | Field 3 |

| 3 | 4 | 6 |

| 8 | 1 | 7 |

| 8 | 8 | 1 |

| 7 | 6 | 4 |

| 1 | 5 | 1 |

Sort:

| Field 1 | Field 2 | Field 3 |

| 7 | 6 | 4 |

| 8 | 8 | 1 |

| 8 | 1 | 7 |

| 3 | 4 | 6 |

| 1 | 5 | 1 |

0 0 4

8 8 1

----------

8 8 5

0 0 5

8 1 7

----------

8 1 12

0 1 5

3 4 6

----------

3 5 11

3 1 5

1 5 1

----------

4 6 6

4 1 5

Max number = 5

СОРТИРОВКА ПО ВОЗРАСТАНИЮ

Matrix:

| Field 1 | Field 2 | Field 3 |

| 3 | 4 | 6 |

| 8 | 1 | 7 |

| 8 | 8 | 1 |

| 7 | 6 | 4 |

| 1 | 5 | 1 |

Sort:

| Field 1 | Field 2 | Field 3 |

| 1 | 5 | 1 |

| 3 | 4 | 6 |

| 8 | 1 | 7 |

| 8 | 8 | 1 |

| 7 | 6 | 4 |

1 0 0

3 4 6

----------

4 4 6

4 0 0

8 1 7

----------

12 1 7

4 1 0

8 8 1

----------

12 9 1

4 1 1

7 6 4

----------

11 7 5

4 1 5

Max number = 5

БЕЗ СОРТИРОВКИ

| Field 1 | Field 2 | Field 3 |

| 3 | 4 | 6 |

| 8 | 1 | 7 |

| 8 | 8 | 1 |

| 7 | 6 | 4 |

| 1 | 5 | 1 |

3 0 0

8 1 7

----------

11 1 7

3 1 0

8 8 1

----------

11 9 1

3 1 1

7 6 4

----------

10 7 5

3 1 5

1 5 1

----------

4 6 6

4 1 5

Max number = 5

# Код программы

# #include <iostream>

# #include <random>

# #include <algorithm>

# #include <vector>

# #include <numeric>

# #include <iomanip>

# #include <iterator>

# #include <chrono>

# #include <functional>

# #include <map>

# #include "prettytable.h"

# int main()

# {

# prettytable::PrettyTable mytable, mytable2;

# int m, n, T1, T2;

# std::vector<int> mas;

# std::cin >> m >> n >> T1 >> T2;

# while (mas.size() < m \* n) {

# mas.push\_back(rand() % (T2 - T1 + 1) + T1);

# }

# std::vector<std::vector<int>> splits(m, std::vector<int>(n));

# std::size\_t offset = 0;

# for (auto& split : splits) {

# auto begin = std::next(mas.begin(), offset);

# auto end = std::next(mas.begin(), offset + n);

# std::copy(begin, end, split.begin());

# offset += n;

# }

# std::vector<std::vector<int>> splits2(splits);

# std::vector<std::vector<int>> splits3(splits);

# std::cout << "\n---СОРТИРОВКА ПО УБЫВАНИЮ---\nMatrix:\n";

# for (const auto& split : splits) {

# mytable.add\_row(split);

# }

# std::cout << mytable << std::endl;

# for (int i = 0; i < m - 1; ++i) {

# for (int j = 0; j < m - i - 1; ++j) {

# if (std::accumulate(splits[j].begin(), splits[j].end(), 0) >

# std::accumulate(splits[j + 1].begin(), splits[j + 1].end(), 0)) {

# std::swap(splits[j], splits[j + 1]);

# }

# }

# }

# std::reverse(splits.begin(), splits.end());

# mytable2.set\_field\_names({});

# std::cout << "Sort:\n";

# for (const auto& split : splits) {

# mytable2.add\_row(split);

# }

# std::cout << mytable2 << std::endl;

# std::vector<std::vector<int>> mass(m, std::vector<int>(n));

# std::vector<int> acc(n);

# for (std::size\_t i = 0; i < splits.size(); ++i) {

# std::copy(splits[i].begin(), splits[i].end(), acc.begin());

# std::copy(acc.begin(), acc.end(), mass[i].begin());

# }

# std::vector<int> masSum(n), masNumbers(n);

# auto min\_it = std::min\_element(mass[0].begin(), mass[0].end());

# auto min\_index = std::distance(mass[0].begin(), min\_it);

# masSum[min\_index] += \*min\_it;

# masNumbers[min\_index] += \*min\_it;

# std::cout << "\n" << std::setw(4) << masSum[min\_index];

# for (const auto& value : mass[0]) {

# std::cout << std::setw(4) << value;

# }

# std::cout << "\n";

# for (std::size\_t i = 1; i < mass.size(); ++i) {

# std::copy(masSum.begin(), masSum.end(), masNumbers.begin());

# masSum = std

# Блок схема

****

# Вывод

Алгоритм построения расписания с произвольной загрузкой достаточно эффективен по скорости поиска приемлемого по точности решения.

# Литература

1. Коффман Э.Г. “Теория расписания и вычислительные машины” – M.: “Наука”, 1987
2. Романовский И.В. “Алгоритмы решения экстремальных задач” – М.: “Наука”, 1977
3. Пашкеев С.Д., Минязов Р.И., Могилевский В.Д. “Машинные методы оптимизации в технике связи” – М.: “Связь”, 1976.