

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

**Лабораторная работа №2**

по дисциплине «Основы теории принятия решений»

**Выполнила:**

студентка уч. гр. ВМО41

Волкова Эмилия Юрьевна

**Проверил:**

проф. Кобак В. Г.

Ростов-на-Дону

2025

**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.

**Алгоритмы распределения работ на параллельно работающие устройства**

Описанный ниже метод более эффективен по скорости поиска приемлемого по точности решения.

Алгоритм построения расписания с произвольной загрузкой:

Дана прямоугольная матрица .

Ш.1 Упорядочим строки матрицы *T* по убыванию сумм всех их элементов.

Ш.2 В преобразованной матрице *T’* первой строке и найдем в ней минимальный элемент. Примем этот элемент за элемент распределения и прибавим его к соответствующему элементу следующей строки.

Ш.3 Следующая строка теперь учитывает предыдущее решение. Выберем из нее минимальный элемент, прибавим его к соответствующему элементу третьей строки и т.д.

A diagram of a flowchart

Description automatically generated

Рисунок 1 – Схема алгоритма построения расписания с произвольной загрузкой

**Вычислительные эксперименты**

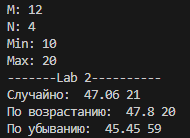
****

Рисунок 2 – Эксперимент №1

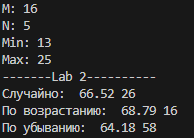


Рисунок 3– Эксперимент №2

**Листинг программы**

import numpy as np

import random

def schedule\_with\_arbitrary\_load(T, N, M):

load = [0 for i in range(N)]

for i in T:

min\_el = min(i[:N])

min\_index = i.index(min\_el)

if load[min\_index] == 0:

load[min\_index] += min\_el

elif load[min\_index] > 0:

load\_min\_index = load.index(min(load))

load[load\_min\_index] += i[load\_min\_index]

return load, max(load)

def sorting(N, T, sort\_arg = -1):

if sort\_arg == -1:

return T

elif sort\_arg == 0:

return sorted(T, key=lambda x: x[N])

elif sort\_arg == 1:

return sorted(T, key=lambda x: x[N], reverse=True)

# Основная программа

def main():

num\_tasks = int(input("M: ")) # количество заданий

processors = int(input("N: ")) # количество процессоров

rand\_min = int(input("Min: "))

rand\_max = int(input("Max: "))

num\_massives = 100

all\_schedule = []

all\_schedule\_asc = []

all\_schedule\_desc = []

schedule\_rand = 0

schedule\_asc = 0

schedule\_desc = 0

for i in range(num\_massives):

tasks = np.random.randint(rand\_min, rand\_max, (num\_tasks, processors)).tolist()

for i in tasks:

i.append(sum(i))

tasks\_asc = sorting(processors, tasks, 0)

tasks\_desc = sorting(processors, tasks, 1)

result\_schedule, max\_rand = schedule\_with\_arbitrary\_load(tasks, processors, num\_tasks)

result\_schedule\_asc, max\_asc = schedule\_with\_arbitrary\_load(tasks\_asc, processors, num\_tasks)

result\_schedule\_desc, max\_desc = schedule\_with\_arbitrary\_load(tasks\_desc, processors, num\_tasks)

all\_schedule.append(max\_rand)

all\_schedule\_asc.append(max\_asc)

all\_schedule\_desc.append(max\_desc)

max\_val\_schedule = min([max\_rand, max\_asc, max\_desc])

if max\_val\_schedule == max\_desc:

schedule\_desc += 1

elif max\_val\_schedule == max\_rand:

schedule\_rand += 1

elif max\_val\_schedule == max\_asc:

schedule\_asc += 1

for i in [all\_schedule, all\_schedule\_asc, all\_schedule\_desc]:

aver = sum(i)/len(i)

i.append(aver)

print("-------Lab 2----------")

print("Случайно: ", all\_schedule[len(all\_schedule)-1], schedule\_rand)

print("По возрастанию: ", all\_schedule\_asc[len(all\_schedule\_asc)-1], schedule\_asc)

print("По убыванию: ", all\_schedule\_desc[len(all\_schedule\_desc)-1], schedule\_desc)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Вывод**

Проанализировав данные, полученные входе экспериментов, можно сделать вывод, что алгоритм построения расписания с произвольной загрузкой более эффективен по скорости поиска подходящего по точности решения.

Литература

1. Поспелов Д.А. “Введение в теорию вычислительных машин” – M.: “Советское радио”, 1972

2. Пашкеев С.Д. “Основы мультипрограммирования для специализированных вычислительных машин” – M.: “Советское радио”, 1964

3. Плотников В.Н., Зверев В.Ю. “Техническая кибернетика №3” M., 1974

4. Бондаренко А.Т., Сапатый П.С. “Техническая кибернетика №4” –Киев, 1975