## Разработка и реализация алгоритма решения прикладной задачи и проведение его экспериментального исследования

Содержательная постановка задачи.
*Дано:*

- N независимых работ;
- для каждой работы задано время выполнения.

*Требуется:* построить расписание выполнения работ без прерываний на M процессорах.

*Алгоритм:* муравьиный алгоритм.

Математическая постановка задачи.
*Дано:*

- **M** - число процессоров;
- **N** - число независимых работ;
- **D[i] ∀ i ∈ [1..N]** - время выполнения *i-ой* работы на процессоре.

*Найти:* такое расписание выполнения работ без прерываний на M процессорах, что:
**D[schedule] = max** $\sum$ **D[j]**, где $j \in [1..N]$ и выполняется на *i-ом* процессоре, - минимально среди всех возможных вариаций построения заданного расписания.

*Представление решения:*
словарь, ключи которого представляют собой номера процессоров (*i*, где $i \in [1..M]$), а значения являются множествами номеров исполняемых на некотором (*i-ом*) процессоре задач (множество $\{j\}$, где $j \in [1..N]$).

Алгоритм решения задачи:
(подразумевается, что читатель знаком с базовым муравьиным алгоритмом)

*Замечание.*
Граф, для которого применяется муравьиный алгоритм в такой постановке задачи, выглядит следующим образом:

- имеется два типа вершин: вершины, соответствующие независимым работам, и вершины, соответствующие процессорам;
- граф ориентированный: из каждой вершины, соответствующей некоторой работе, исходят дуги во все вершины, соответствующие процессорам; и наоборот, из каждой вершины, соответствующей некоторому процессору, исходят дуги во все вершины, соответствующие работам.

1. Инициализируем коэффициент испарения феромона (*p = 0.05*), коэффициенты для подсчета вероятностей: *α* и *β (alpha = 1.5; beta = 2), α ≤ β*.
2. Принимаем за текущий лучший результат работы алгоритма (условно минимальное для всех возможных вариаций расписаний максимальное время работы среди процессоров) сумму всех заданных длительностей выполнения работ.
3. Принимаем количество феромона (т) на всех ребрах равным некоторому небольшому числу (*ph = 0.05*). Локальная целевая функция (η) на ребрах, идущих от вершины работы к вершине процессора, принимаем равной 1, а на ребрах, идущих от вершины процессора к вершине работы, - равной времени выполнения этой работы.
4. Количество муравьев принимаем равным количеству работ; сажаем на каждую вершину, соответствующую некоторой работе, по одному муравью.
5. Для каждого муравья строим по муравьиному алгоритму маршрут в графе, в котором муравей посещает все вершины, соответствующие работам. Когда муравей находится в вершине, соответствующей работе, он выбирает следующей вершину, соответствующую процессору; и наоборот. Работа считается размещенной на процессоре, если в маршрут муравья входит дуга, соединяющая вершину этой работы с вершиной этого процессора (и направленная от вершины работы к вершине процессора).
6. В табу-список муравья в процессе построения пути попадают вершины, соответствующие уже размещенным работам, и вершины, соответствующие процессорам, время выполнения которых превысило текущий лучший результат работы алгоритма.
7. После того, как маршруты всех муравьев построены, строится расписание выполнения работ (используя то, что работа считается размещенной на процессоре, если в маршрут муравья входит дуга, соединяющая вершину этой работы с вершиной этого процессора) и вычисляется максимальное время работы процессора для каждого такого расписания. Находим лучшее (минимальное) такое значение и сравниваем его с уже существующим лучшим результатом, при необходимости обновляем, получая новое лучшее значение.
8. После каждой итерации работы алгоритма феромон на всех ребрах подвергается испарению (с заданным коэффициентом испарения); на ребрах, соответствующих лучшему маршруту, феромон добавляется (причем по разным правилам для ребер, исходящих из вершин процессоров, и ребер, исходящих из вершин работ).
9. *Пункты 4-8* повторяем некоторое заранее заданное количество раз.

*Результатом работы алгоритма является то решение, которое было лучшим, когда все итерации закончились.*

Экспериментальное исследование.

Гипотеза для файла test1.xml: максимальное время работы процессора D: 150≤D≤165.

Гипотеза подтверждена экспериментально.

При количестве итераций 1000 алгоритм стабилизируется и возвращает значение:

D = 152.

Пример результатов для запусков с 1000 итераций:

```
163
163
...
160
160
160
160
...
152
152
152
```

Единственный столбец здесь - максимальное время работы процессора на очередной итерации.

Выводы.

Алгоритм находит оптимальное решение с некоторой вероятностью, которая зависит от количества итераций алгоритма и от размерности входных данных.

Приложение.

Файл generator.py содержит генератор входных данных. При использовании в аргументах командной строки требуется указать n - количество процессоров, m - количество задач и файл, в который будут записаны сгенерированные данные.

*Формат представления входных данных.*

В корневом элементе *want_schedule* с атрибутами *number_of_processors* (число процессоров) и *number_of_tasks* (число работ) содержится список из *number_of_tasks* элементов *task* (очередной работы) с атрибутом *task_number* (номер этой работы). Подэлемент *duration_time* отражает длительность этой работы.

*Представление данных в основной программе.*
Число процессоров и число работ представлены переменными *processors* и *tasks.* Время выполнения каждой работы отражено в списке *time_tasks*.