1. Условия задачи

Дан набор вычислительных задач и набор виртуальных машин. Для каждой задачи известен объём вычислений и требования к памяти, а для каждой виртуальной машины — производительность и ограничения по памяти. Для задач заданы отношения предшествования (вида «задача C должна выполняться после задач A и B»).

Необходимо назначить каждую задачу одной виртуальной машине так, чтобы:
- все ограничения по ресурсам были выполнены;
- все отношения предшествования были соблюдены;
- общее время выполнения всех задач (makespan) было минимальным.

2. Формализация задачи

2.1. Структура решения

Пусть n — количество задач, m — количество виртуальных машин.

Для каждой задачи $i = 1, ..., n$ заданы:
- T_i — объём вычислений (в условных единицах),
- V_i — требуемый объём памяти,
- $P_i \subseteq \{1, ..., n\}$ — множество непосредственных предшественников (не содержит циклов).

Для каждой виртуальной машины $j = 1, ..., m$ заданы:
- F_j — производительность (единиц вычислений в единицу времени),
- M_j — доступный объём памяти.

Решение представляет собой назначение каждой задачи одной машине и определение порядка выполнения задач на каждой машине (должен быть совместим с отношениями предшествования).

2.1.1. Представление задачи на графах

Задача планирования вычислительных задач с учётом предшествований и ограничений ресурсов может быть формализована с использованием ориентированного ациклического графа $G = (V, E)$, где $V = \{v_1, v_2, ..., v_n\}$ — множество вершин, каждая из которых представляет задачу $i = 1, ..., n$, аннотированную атрибутами T_i (объём вычислений) и V_i (требуемый объём памяти). Множество рёбер $E \subseteq V \times V$ определяет отношения предшествования: ребро $(v_k, v_i) \in E$ существует, если задача k является непосредственным предшественником задачи i (т.е., $k \in P_i$), обеспечивая отсутствие циклов в графе.

Для интеграции виртуальных машин в модель вводится дополнительное множество вершин $U = \{u_1, u_2, ..., u_m\}$, где каждая вершина u_j соответствует виртуальной машине $j = 1, ..., m$ с атрибутами F_j (производительность) и M_j (доступная память). Назначение задач машинам эквивалентно добавлению рёбер между вершинами V и U, образуя двудольный граф $H = (V \cup U, E_H)$, где E_H — множество рёбер назначения вида (v_i, u_j), указывающее, что задача i выполняется на машине j. Каждое такое ребро должно удовлетворять ограничению по памяти: $V_i \leq M_j$. Порядок выполнения задач на одной машине определяется топологическим порядком в подграфе, индуцированном задачами, назначенными на эту машину, с учётом исходных предшествований в G.

Время выполнения задачи i на машине j вычисляется как $t_i^j = \frac{T_i}{F_j}$, а момент начала S_i — как максимум из моментов завершения предшественников и предыдущих задач на той же машине: $S_i = \max\Big\{\max_{k \in P_i} C_k, \max\{C_l \mid l \text{ предшествует } i \text{ на машине } j\}\Big\}$, где $C_i = S_i + t_i^j$.

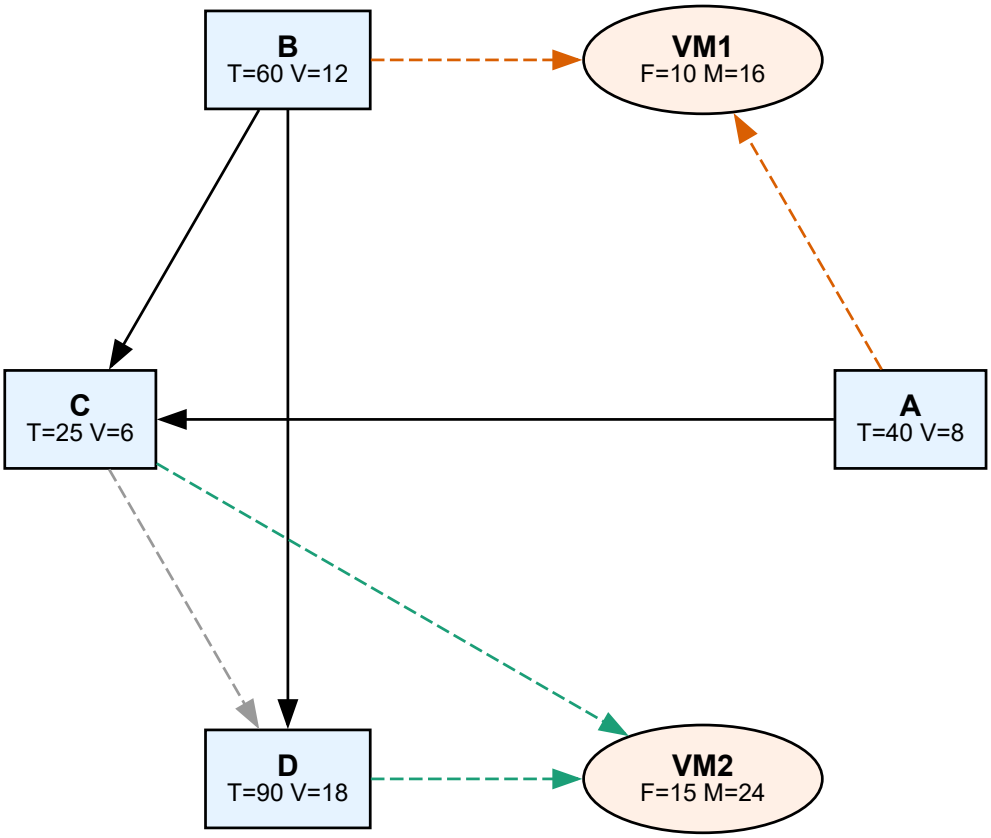

Рис. 1. Пример графа

2.1.2. Способы кодирования решения

1. Вектор назначения

$$X = (x_1, x_2, ..., x_n), \quad x_i \in \{1, 2, ..., m\}$$

где $x_i = j$ означает, что задача i назначена на машину j.

2. Булева матрица назначения

$$M_{ij} \in \{0, 1\}, \quad \sum_{j=1}^{m} M_{ij} = 1 \quad \forall i \in 1..n$$

2.2. Целевая функция

Цель — минимизация общего времени завершения всех задач:

$$f(X) = \max_{i=1..n} C_i$$

где

- C_i — момент завершения задачи i,
- $C_i = S_i + t_i^{x_i}$,

- $t_i^j = \frac{T_i}{F_j}$ — время выполнения задачи i на машине j,
- S_i — момент начала выполнения задачи i.

Значение S_i определяется как

$$S_i = \max\left\{\max_{k \in P_i} C_k, \quad \max\{C_k : \text{k выполняется на той же машине перед i}\}\right\}$$

2.3. Способ учёта ограничений

Ограничения задачи делятся на два типа:
- жёсткое соблюдение.
- мягкое учёт через штрафные функции.

Жёсткие ограничения (должны выполняться в любом допустимом решении):

1. Каждой задаче назначается ровно одна виртуальная машина:

$$x_i \in \{1, ..., m\} \quad \forall i = 1..n$$

2. Ограничение по памяти для каждой задачи:

$$V_i \le M_{x_i} \quad \forall i$$

3. Соблюдение порядка предшествования:

$$S_i \ge \max_{k \in P_i} C_k \quad \forall i \quad \text{с непустым } P_i$$

Варианты штрафных функций для мягкого учёта ограничений:

- Штраф за нарушение памяти по каждой задаче:

$$p_i^{\text{mem}} = \max\left(0, V_i - M_{x_i}\right) \cdot K_{\text{mem}}$$

- Штраф за нарушение предшествования:

$$p_i^{\text{pred}} = \max\left(0, \max_{k \in P_i} C_k - S_i\right) \cdot K_{\text{pred}}$$

- Общий штраф решения:

$$P(X) = \sum_{i=1}^{n} \left(p_i^{\text{mem}} + p_i^{\text{pred}}\right)$$

Итоговая функция, которую минимизирует алгоритм:

$$\tilde{f}(X) = \max_{i=1..n} C_i + P(X)$$