Задача об оптимальном расписании (случай идентичных машин)

Identical-machines scheduling problem, IMS

Александр Стешенко
ФПМИ МФТИ

12.11.2023

Аннотация

В данной работе рассматривается задача IMS, Identical-Machine Scheduling, являющаяся частным случаем задачи об оптимальном расписании — одной из важнейших **NP**-полных задач. Суть задачи заключается в нахождении оптимальной последовательности выполнения некоторого множества задач с разным временем выполнения на нескольких идентичных машинах. В ходе работы будет доказана **NP**-полнота задачи об оптимальном расписании (и, как следствие, ее частного случая IMS). Также будет построен жадный алгоритм Грэма, дающий $\frac{4}{3}$-приближение для задачи IMS, проанализированный Рональдом Грэмом в 1960-х для решения IMS и смежных ей задач.

Содержание

1 Введение

1.1 Общая информация

Задача об оптимальном расписании на идентичных машинах (процессорах, исполнителях) — один из вариантов задачи об оптимальном расписании. В задаче об оптимальном расписании в общем случае требуется найти последовательность исполнения, позволяющую выполнить все задачи за минимальное время, причем каждая машина имеет свою производительность, а каждая задача - свою трудоемкость. В случае IMS рассматриваются идентичные, то есть имеющие одинаковую производительность, машины, но задачи разной трудоемкости.

Обозначим явно следующие особенности рассматриваемой проблемы:

1. Каждая машина выполняет только одну задачу в один момент времени;
2. Каждая задача должна выполняться только на одной машине и без прерываний;
3. Смена задач происходит мгновенно;
4. Производительность машин постоянна, а трудоемкость задач известна заранее;
5. Задачи не зависят друг от друга и могут быть выполнены в любом порядке.

Если зафиксировать также ограничение k на итоговое время работы, то задачу распознавания ситуации, в которой можно составить корректное расписание, можно рассматривать **NP**-полную, что будет доказано ниже.

1.2 Область применения

Задача IMS является ключом к многим повседневным задачам. Помимо очевидных приложений в ускорении работы производств, больниц и планирования работы команд, задача IMS может быть применена в оптимизации распределённых вычислений и распредлённого хранения, вычислений на видеокартах, планировщиков операционных систем и множестве других задач.

2 Теория

2.1 Формальная постановка задачи

Будем использовать следующие обозначения:

1. M — машины, $|M| = m$;
2. J — задачи, $|J| = n$;
3. $p : J \times M \longrightarrow \mathbb{R}_+$, где p_{ij} — время выполнения задачи i на машине j.

Так как все машины имеют в нашем случае одинаковую производительность, будем в дальнейшем считать $p : J \longrightarrow \mathbb{R}_+$, где p_i - время выполнения задачи i на любой машине.

Определение 2.1. Тогда задача IMS заключается в нахождении такой $x : J \times M \longrightarrow \{0, 1\}$ (расписания), что:

1. $$\max_{j \in M} \sum_i x_{ij} p_i \longrightarrow min \ =: \ T_{min}(J, M, p)$$
2. $$\forall i \sum_{j \in M} x_{ij} = 1$$

2.2 Необходимые определения

Определение 2.2. Детерминированной машиной Тьюринга с k лентами называется кортеж $\langle \Sigma, \Gamma, Q, q_1, q_a, q_r, k, \delta >$, где Σ, Γ, Q — конечные непустые множества, и $\Sigma \subset \Gamma, \Gamma \cap Q =, q_1, q_a, q_r \in Q$ попарно различны, а δ есть функция из $(Q \setminus \{q_a, q_r\}) \times \Gamma^k$ в $Q \times \Gamma^k \times \{L, N, R\}^k$. Множество Σ называется входным алфавитом, Γ — ленточным алфавитом, Q — множеством состояний, q_1, q_a, q_r — начальным, принимающим и отвергающим состояниями соответственно, а δ — функцией перехода. Среди элементов Γ выделяют специальный символ #, не входящий в множество Σ.

Определение 2.3. Машина распознаёт язык A за время $T(n)$, если она принимает все слова, лежащие в A, отвергает все слова, не лежащие в A, и на каждом слове x работает не больше $T(|x|)$ шагов.

Определение 2.4. Классом $\mathbf{DTIME}(T(n))$ называется класс языков, которые распознаются за время $O(T(n))$. Иными словами, время работы машины на любом слове длины n не превосходит некоторой константы, умноженной на $T(n)$.

Определение 2.5. $\mathbf{P} = \bigcup_{c=1}^{\infty} \mathbf{DTIME}(n^c)$.

Определение 2.6. Классом $\mathbf{NTIME}(T(n))$ называется множество языков, распознаваемых на недетерминированной машине Тьюринга за время $O(T(n))$.

Определение 2.7. $\mathbf{NP} = \bigcup_{c=1}^{\infty} \mathbf{NTIME}(n^c)$.

Альтернативное определение:

Определение 2.8. Классом **NP** называется множество языков A, для которых существует функция $V(x, s)$ с булевыми значениями, вычислимая за полиномиальное время от длины первого аргумента, такая что:

1. Если $x \in A$, то $\exists s : V(x, s) = 1$;
2. Если $x \notin A$, то $\forall s \hookrightarrow V(x, s) = 0$.

Определение 2.9. Пусть A и B суть два языка. Тогда A сводится по Карпу к B, если существует всюду определённая функция $f : \{0,1\}^* \to \{0,1\}^*$, вычислимая за полиномиальное время, такая что $x \in A \Leftrightarrow f(x) \in B$ Обозначение: $A \leqslant_p B$.

Определение 2.10. Язык B является **NP**-трудным, если для любого $A \in \mathbf{NP}$ выполнено $A \leqslant_p B$. Язык B является **NP**-полным, если он **NP**-трудный и лежит в **NP**.

3 NP-полнота IMS с ограничением k на время работы

Напомним, что задача минимизации не может лежать или быть полной в каком-то сложностном классе; в нем может лежать задача распознавания. Необходимо явно задать язык, с которым будем работать, установив ограничение на возможные решения. В нашем случае ограничим общее время работы некоторой константой k.

Определение 3.1. $\mathsf{IMS_k} = \{\langle J, M, p\rangle \mid T_{min}(J, M, p) < k\}$

Таким образом, задача отыскания оптимального расписания не проще, чем распознавание языка $\mathsf{IMS_k}$, т.к. имея оптимальное расписание для некоторой тройки $\langle J, M, p\rangle$ можно за полиномиальное время найти $T_{min}(J, M, p)$ и тем самым однозначно ответить на вопрос ее принадлежности языку $\mathsf{IMS_k}$ $\forall k$.

Согласно определению 2.9, для доказательства **NP**-полноты $\mathsf{IMS_k}$ будет достаточно доказать:

1. $\mathsf{IMS_k} \in \mathbf{NP}$;
2. $\mathsf{IMS_k}$ является **NP**-трудным.

Теорема 3.2. $\mathsf{IMS_k} \in \mathbf{NP}$

Доказательство. Воспользуемся сертификатным определнием 2.7 принадлежности языка **NP**. Рассмотрим функцию $V(x, s)$, и сертификат s — фукнцию x из 2.1. При записи на ленте s можно представить как булевую матрицу размера $n \times m$, где $s[i][j] = 1 \Leftrightarrow$ задачу i будет выполнять машина j.Тогда машина Тьюринга, вычисляющая $V(x, s)$, должна будет проверить следующие условия:

1. $$\forall i \sum_{j \in M} s_{ij} = 1$$
2. $$\max_{j \in M} \sum_{i} s_{ij} p_i \longrightarrow t,\ t \leq k$$

Оба условия могут быть проверены за полиномиальное время, т.к. требуется лишь вычислить две линейные от размера входа суммы для каждого из линейного числа входных значений и сравнить значения с константами. Таким образом, $\mathsf{IMS}_k \in \mathbf{NP}$. $\square$

Определение 3.3.

$$\mathsf{SUBSETSUM\ (SSP)} = \{\langle S = \{s_1, s_2, ..., s_n\}, q\rangle \mid \exists S' \subseteq S : \sum_{s \in S'} = q\}$$

Утверждение 3.4. SSP является **NP** - полным языком.
Данный факт примем общеизвестным

Теорема 3.5. $\mathsf{IMS_k}$ является **NP**-трудным.

Доказательство. Согласно 2.9 будет достаточно показать, что известный **NP**-полный язык $\mathsf{SSP} \leqslant_p \mathsf{IMS_k}$. Сведем решение задачи B из класса SSP к $\mathsf{IMS_k}$, построив f следующим образом:

1. f считает $\sum_{i=1}^{n} s_i = Q$ за линейное время
2. Сравнивает Q и $2q$ за константное время
3. (a) Если $Q < 2q$ рассматривает $A := \mathsf{IMS_q}(S \cup \{2q - Q\}, 2, S \cup \{2q - Q\})$
   (b) Если $Q > 2q$ рассматривает $A := \mathsf{IMS_q}(S \cup \{Q - 2q\}, 2, S \cup \{Q - 2q\})$
   (c) Если $Q = 2q$ рассматривает $A := \mathsf{IMS_q}(S, 2, S)$

   Добавление задачи во множество позволяет максимально сравнять время работы двух машин. Переход линеен, $A \in \mathsf{IMS_k} \Leftrightarrow B \in \mathsf{SSP}$

Получили задачу из класса $\mathsf{IMS_k}$ за полиномиальное время. Решение исходной и получившихся задач будет эквивалентно, так как в любом случае имеем $Q' = \sum_{i=1}^{n+1} s_i = 2q$, и если в исходном множестве можно было выделить подмножество суммы q, то при решении $\mathsf{IMS_k}$ передадим на первую машину это подмножество, а на другую - остальные элементы S, и $T_{min} = q$. Если же в исходном множестве такого подмножества не было, то не будет ни одного способа резделить задачи между машинами поровну и, учитывая $Q' = 2q$ одна из машин привысит время работы q. □

Теорема 3.6. $\mathsf{IMS_k}$ является **NP**-полным.

Доказательство. Согласно 3.2 $\mathsf{IMS_k} \in \mathbf{NP}$, а согласно 3.5 — $\mathsf{IMS_k}$ является **NP**-трудным. Тогда по определнию получаем, что $\mathsf{IMS_k}$ **NP**-полон. □

4 Алгоритм Грэма для $\frac{4}{3}$-приближения решения IMS

4.1 Описание алгоритма

Жадный алгоритм, предложенный Рональдом Грэмом для приближения задачи IMS за полиномиальное время в 1969 году:

1. Отсортируем задачи по убыванию их трудоемкости;
2. Каждая машина, освбождаясь, выбирает первую доступную в списке задачу, начинает выполнять ее и удаляет ее из списка. Если свободны одновременноо две машины, машина с меньшим индексом берет задачу с меньшим индексом. Если доступных задач в списке нет, машина простаивает;
3. Алгоритм завершается, если в списке не осталось доступных задач, и все машины простаивают;
4. Приближенное время работы находится как максимум по времени работы всех машин.

Напомним, что решаем задачу в статическом случае, то есть когда новых задач не появляется после начала работы.

4.2 Доказательство приближения

Теорема 4.1. Обозначим результат работы приведенного алгоритма как T', а оптимальное значение - T_{min}, m - количество машин. Тогда имеет место:

$$1 \leq \frac{T'}{T_{min}} \leq \frac{4}{3} - \frac{1}{3m}$$

Доказательство. 1. $1 \leq \frac{T'}{T_{min}}$ очевидно в силу оптимальности T_{min}.

2. Приведено в работе Грэма на страницах 422-426

□

4.3 Псевдокод

Graham approximation for IMS problem

Require: $n \geq 0,\ m \geq 0,\ J = \{J_1 \geq 0, ..., J_n \geq 0\}$
Ensure: $T_{min}(J, m) \leq T'(J, m) \leq \frac{4}{3}(T_{min}(J, m))$
  $M_1, ..., M_m \leftarrow 0$
  $sort - decrease - order(J)$
  $i \leftarrow 0$
  while $i < n$ **do**
    $j = argmin(M_1, ..., M_m)$
    $M_j = M_j + J_i$
    $i = i + 1$
  end while
  $T' \leftarrow max(M_1, ..., M_m)$

Итого имеем алгоритмическую сложность $O(nlogn + nlogm) = O(n(logn + logm))$ и укладываемся в полином. При этом алгоритм, основанный на переборе, имеет асимптотику $O(m^n)$.

4.4 Реализация на языке C++

Приведена на странице проекта.

4.5 Анализ результатов

Были проведены две серии тестов: сравнивающие время работы наивного алгоритма и алгоритма Грэма, а также проверяющие степень приближения ответа алгоритмом Грэма. Сравнение времени на приближения задачи IMS алгоритмом Грэма и точного ее решения на случайных задачах представлено ниже:

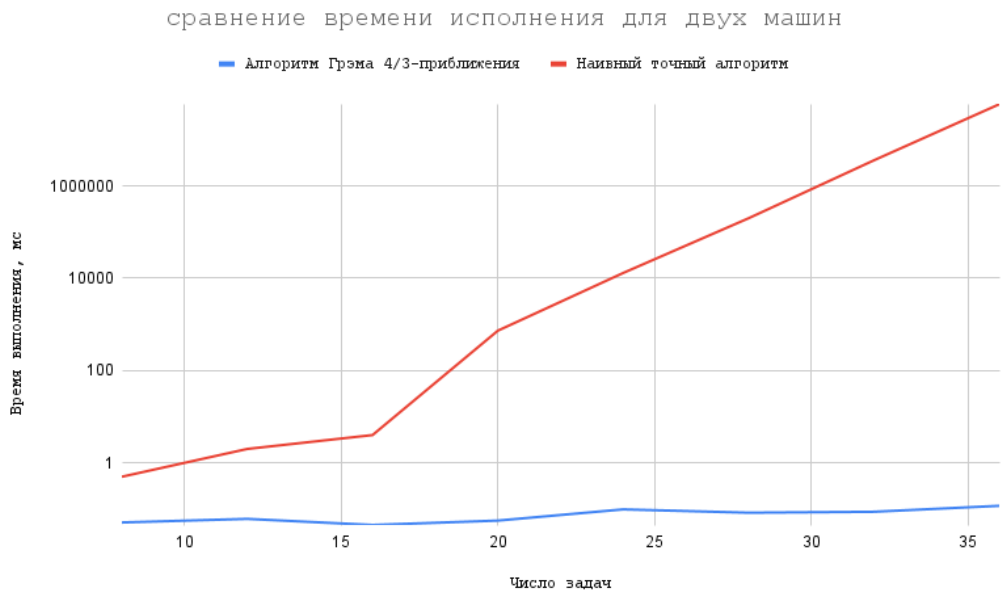

Рис. 1: данные графика

Видно, что наивный алгоритм приближения имеет экспоненциальную скорость работы, в то время как приближение Грэма - субэкспоненциальную. Даже на двух машинах и нескольких десятках задач наивный алгоритм становится практически не применим, в то время как построенный алгоритм даже не меняет продолжительность своей работы вне пределов погрешности. Проверим выводы на наборе из трёх машин:

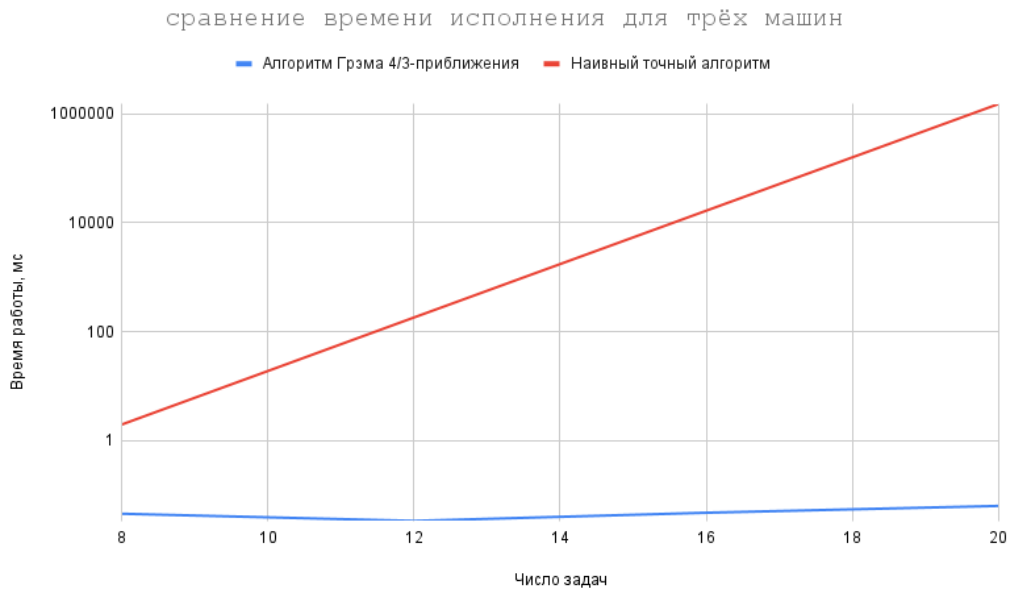

Рис. 2: данные графика

Можем сделать те же самые выводы. Для числа машин больше 4 время работы наивного алгоритма становится недопустимо большим, а потому для таких наборов входных данных сравнение времени работы опустим.

Тестирование приближения решения IMS на 100 тестах подтвердило теоретическую оценку приближения $\frac{4}{3} - \frac{1}{3m}$.

5 Заключение

Предложенный Р. Л. Грэмом алгоритм для задачи IMS весьма эффективен, ведь позволяет со сложностью $O(n(logn + logm))$ с точностью до множителя $\frac{4}{3}$ приблизить **NP**-полную задачу. Уже при небольших входных данных время работы алгоритма, основанного на полном переборе становится непозволительно долгим, что лишает всякого смысла нахождение точного решения задачи IMS при $\mathbf{P} \neq \mathbf{NP}$. В то же время точность $\frac{4}{3}$ является приемлемой при решении большинства практических задач.

6 Литература

1. Bounds on Multiprocessing Timing Anomalies, R. L. Graham;

2. Сложность вычислений, конспект лекций, Д.В. Мусатов.