Московский физико-технический институт (ГУ)
Физтех-школа прикладной математики и информатики
Сложность вычислений, весна 2023

Задача об оптимальном расписании для случая идентичных исполнителей. Алгоритм $\frac{4}{3}$-аппроксимации и его анализ

Смирнов Владислав
Б05-121

Долгопрудный, 2023

Аннотация

Задача об оптимальном назначении задач является фундаментальной в области комбинаторных оптимизаций, она регулярно появляется в реальных прикладных задачах. Одной из самых значимых фигур в области теории расписания является американский математик Рональд Грэм, который внёс поистине большой вклад в её развитие. Автор статьи провёл исследование данной задачи для случая идентичных исполнителей задач (при условии упреждающего планирования). Доказана **NP**-полнота соответствующей задачи, построен и имплементирован алгоритм $\frac{4}{3}$-приближения.

Содержание

1 Введение

1.1 Постановка задачи

Имеется множество работ J и множество машин M. Также задана функция $p : J \times M \to \mathbb{R}_+$. Значение p_{ij} означает время выполнения i-й работы на j-й машине. Требуется построить распределение работ по машинам так, чтобы все работы были выполнены и чтобы конечное время выполнения всех работ было минимально. Иначе говоря, требуется найти функцию $x : J \times M \to \{0, 1\}$ такую, что:

$$\max_{j \in M} \sum_i x_{ij} p_{ij} \to min$$

$$s.t. \sum_{j \in M} x_{ij} = 1$$

1.2 Формализация задачи

В нотации Рональда Грэма поставленная задача обозначается через $P||C_{max}$, где P обозначает идентичных исполнителей и C_{max} – целевую функцию, которую мы минимизируем. Пусть $|M| = m, |J| = n$. Каждая машина в каждый момент времени решает не более чем одну задачу (работу). Как только машина начинает выаполнять i-ю работу, через время p_i она её завершает и становится готовой выполнять следующую. Обозначим через J_j множество работ, назначенных на j-ю машину. Таким образом, расписание задаётся набором подмножеств множества данных работ $\{J_1, J_2, \ldots, J_m\}$. Обозначим через

$$C(J_j) = \sum_{i \in J_j} p_i$$

нагрузку j-й машины в соответствии с расписанием. Тогда наша целевая функция определяется как

$$C_{max} = \max_{1 \le j \le m} C(J_j).$$

Далее автор будет использовать термины *задача* и *работа* как равносильные. Будем рассматривать частный случай задачи, когда производительности всех машин одинаковы, то есть $p_{ij} = p_i$.

2 NP-полнота задачи

Поставленная задача может быть переформулирована в проблему принятия решения следующим образом:

Пусть в дополнение к предыдущим входным данным даётся некоторое положительное число K. Существует ли назначение задач на машины, такое что нагрузка на каждую машину не превосходит K?

Докажем, что задача об оптимальном расписании для случая идентичных машин является **NP**-полной, путём сведения её к задаче об упаковке (Bin packing problem). **NP**-полнота задачи об упаковке доказывается преобразованием её из задачи о разделении (Partition problem) (см. [2]). Оригинальную идею доказательства автор данной статьи нашёл в документе [1].

Формулировка задача об упаковке:

Дан некоторый конечный набор элементов U. Вес каждого элемента определяется функцией $s : U \to \mathbb{Z}_+$. Даны два положительных целых числа B и l. Существует ли упаковка элементов множества U в l множеств $U_1, U_2, \ldots, U_l$, такая что суммарный вес каждого из полученных множеств не превосходит B?

Теорема 1. *Поставленная задача* **NP***-полна.*

Доказательство. $P||C_{max} \in \mathbf{NP}$, поскольку недетерминированный алгоритм будет за полиномиальное время проверять, удовлетворяет ли некоторое назначение работ на машины ограничению на суммарное время выполнения K. Сведём нашу задачу к задаче об упаковке: $U := J$, $s(u) := p_i$ (i-я работа – элемент u множества U), $l := |M|$ и $B := K$. Таким образом, **NP**-полнота задачи об оптимальном расписании для случая идентичных машин доказана. □

3 Алгоритм решения задачи и его анализ

3.1 Идея алгоритма

Одним из наиболее известных конструктивных алгоритмов для решения поставленной задачи является жадный алгоритм LPT (Longest-processing-time-first scheduling). Другими популярными алгоритмами аппроксимации поставленной задачи являются List scheduling, MULTIFIT, COMBINE и LISTFIT. Последние три теоретически дают наиболее точную аппроксимацию, но требуют больше вычислительных ресурсов (подробнее про их коэффициент аппроксимации и временную сложность см. в [3]). В то же время на практике алгоритм LPT работает лучше и, кроме того, он прост в реализации.

3.2 Правило LPT

Алгоритм LPT работает следующим образом:

1. Сначала он сортирует работы в порядке убывания времени их выполнения;
2. Далее он назначает работы одну за другой в таком порядке, чтобы очередная работа планировалась на машину с наименьшей в данный момент нагрузкой.

Второй шаг алгоритма по сути является алгоритмом List scheduling. Разница в том, что последний перебирает задачи в произвольном порядке, в то время как LPT предварительно упорядочивает их по времени обработки.

3.3 Коэффициент аппроксимации

LPT был впервые проанализирован Рональдом Грэмом в 1960-х годах в контексте проблемы о назначении задач. Докажем, что достигается $\frac{4}{3}$-приближение. Идею достаточно простого и прямого доказательства автор статьи нашёл в документе [3].

Обозначим через OPT время выполнения в случае оптимального назначения работ для $P||C_{max}$. Разделим множество данных работ J на три группы:

1. Простые, $p_i \leq \frac{1}{3}OPT$
2. Средние, $\frac{1}{3}OPT < p_i \leq \frac{2}{3}OPT$
3. Тяжёлые, $\frac{2}{3}OPT < p_i$

Выделим три этапа работы алгоритма LPT:

1. Сначала m самых длинных работ назначаются на машины. В случае, если $|n| \leq |m|$, алгоритм завершает свою работу

2. Далее алгоритм продолжает назначать задачи, пока очередной выбранной машиной (с наименьшей в данный момент нагрузкой) не станет та, которая выполнила уже две назначенные работы, или пока все задачи не будут назначены

3. Далее алгоритм распределяет оставшиеся задачи, пока все из них не будут назначены

Обозначим значения целевой функции после каждого из описанных выше этапов через C^1_{max}, C^2_{max} и C^3_{max} соответственно. Заметим, что $C^3_{max} = C_{max}$. Также обозначим через Σ расписание, которое генерирует алгоритм LPT и которому соответствует целевая функция C_{max}.

Следующие три леммы позволят нам прийти к требуемуму результату.

Лемма 1. *После завершения первого этапа все работы из группы тяжёлых работ будут назначены. Более того,* $C^1_{max} \leq OPT$.

Доказательство. Заметим, что количество тяжёлых задач не превосходит $|m|$, поскольку иначе значение целевой функции превосходило бы $2 \cdot \frac{2}{3}OPT$. Таким образом, поскольку на каждую машину назначается не более чем одна тяжёлая работа и выполнение каждой работы занимает не более чем OPT времени, верно следующее соотношение: $C^1_{max} \leq OPT$. □

Лемма 2. *После завершения второго этапа все средние работы будут назначены, и если какая-то работа была назначена на машину, на которую уже была назначена тяжёлая работа, то эта задача входит в группу простых задач. Более того,* $C^2_{max} \leq \frac{4}{3}OPT$.

Доказательство. Обозначим через n_L число тяжёлых работ. Отметим следующее:

- В оптимальном решении эти n_L работ назначаются на n_L разных машин.
- В оптимальном решении на машину, которая должна выполнять тяжёлую работу, не может назначаться работа из группы средних.
- Если бы число средних работ превосходило $2(m - n_L)$, любое оптимальное решение требовало бы более чем OPT времени. Таким образом, число средних работ не превосходит $2(m - n_L)$.

На втором этапе $(m - n_L)$ средних работ назначаются на машины, которые не выполняют тяжёлую работу. Каждая из таких машин выполняет не более чем две средние работы. Следовательно, по окончании второго этапа все средние работы будут назначены и, если какая-то задача была назначена на машину, выполняющую тяжёлую работу, то эта задача – простая. Учитывая, что по окончании второго этапа каждая машина выполняет не более двух работ, получаем следующее соотношение: $C^2_{max} \leq \frac{4}{3}OPT$. □

Лемма 3. *После завершения третьего этапа все простые работы будут назначены. Более того,* $C^3_{max} \leq \frac{4}{3}OPT$.

Доказательство. Обозначим через J_i задачу, выполнение которой по распределению задач Σ (расписанию) завершается последним. Обозначим через M_j машину, которая выполняет работу J_i. Заметим, что перед назначением работы J_i на машину M_j нагрузка этой машины составляет не более чем OPT. В противном случае, нагрузка на других машинах тоже превышала бы OPT, поскольку перед каждым назначением выбирается машина с минимальной нагрузкой, а значит, не существовало бы такого распределения задач, на которое сумммарно уходило бы OPT времени. Рассмотрим несколько случаев:

- Если M_j выполняет ровно одну работу, то Σ – оптимальное расписание, поскольку время каждой работы не превосходит OPT, а машина M_j последней завершает свою единственную работу J_i. Таким образом, $C^3_{max} = OPT$.

- Если M_j выполняет ровно две работы, то поскольку это не могут быть две длинные работы или длинная со средней работы, имеем следующее соотношение: $C^3_{max} \leq \frac{4}{3}OPT$.

- Если M_j выполняет более чем две работы, то по второй лемме работа J_i должна быть простой. Поскольку нагрузка машины M_j перед назначением задачи J_i не превосходит OPT, получаем следующее соотношение: $C^3_{max} \leq OPT + p_i \leq \frac{4}{3}OPT$.

□

Теорема 2. *Правило* LPT *даёт алгоритм* $\frac{4}{3}$*-аппроксимации для задачи об оптимальном расписании в случае идентичных исполнителей задач.*

3.4 Анализ работы LPT и List scheduling

Асимптотики обоих алгоритмов совпадают: $O(nlogn + nlogm)$ времени и $O(n + m)$ памяти.

Assign – функция распределения работ на машины. Если передаётся параметр $True$, для распределения используется алгоритм LPT, иначе – алгоритм List scheduling.

Алгоритм, строящий оптимальное расписание, автор реализовал полным перебором (за исключением случая, когда работ меньше, чем машин).

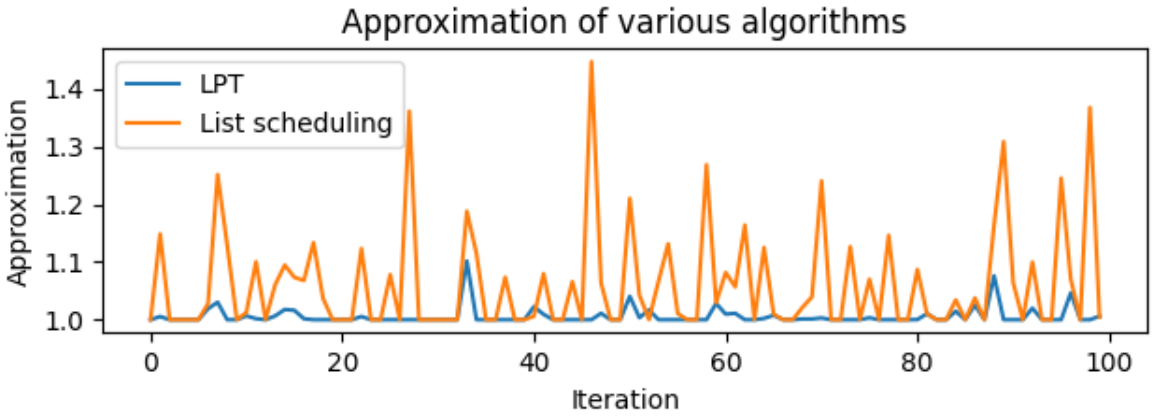
Для построенных алгоритмов написаны тесты, которые проверяют как общие, так и краевые случаи поставленной задачи, а также границы коэффициента аппроксимации. Как отмечается в документе [3], алгоритм List scheduling даёт 2-приближение.

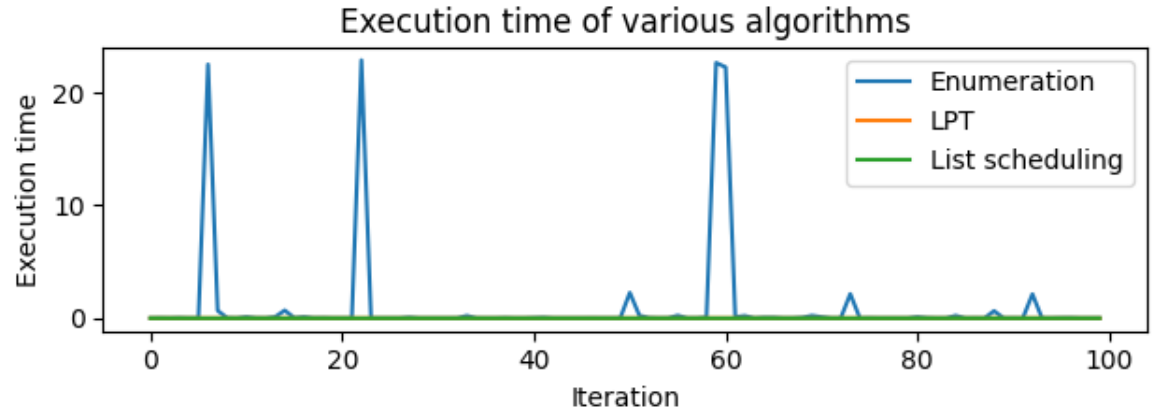
Для проверки корректности приведённых алгоритмов сгенерированы 100 экземпляров поставленной задачи, в которых $1 \leq |J| \leq 12$ и $1 \leq |M| \leq 3$, а время, необходимое на выполнение одной работы не превосходит 10^6. Для каждого из

Algorithm 1 Реализация алгоритмов LPT и List scheduling

```
import bisect


def assign(n, m, Jobs, Schedule, LPT=False):
    if LPT:
        Jobs.sort(reverse=True)

    result = 0
    MachineLoad = list((0, i) for i in range(m))
    for jobCapacity, jobID in Jobs:
        machineLoad, machineID = next(iter(MachineLoad))
        MachineLoad.remove((machineLoad, machineID))
        newMachineLoad = machineLoad + jobCapacity
        Schedule[machineID].append([jobID, newMachineLoad])
        bisect.insort_left(MachineLoad, (newMachineLoad,
                                         machineID))
        result = max(result, newMachineLoad)

    return Schedule, result
```

них посчитано, какое приближение дают алгоритмы LPT и List scheduling, замерено время выполнения всех алгоритмов, включая полный перебор. Построены соответствующие график:





Как видно из первого графика, оценка сверху на коэффициент приближения для обоих алгоритмов не нарушается. Более того, для сгенерированных входных данных алгоритмы LPT и List scheduling дают приближение не хуже $\frac{11}{10}$ и $\frac{3}{2}$ соответственно.

Таблица 1: Итоговая таблица с замерами

	Enumeration	LPT	List scheduling
Average approximation coefficient	1.000	1.010	1.045
Average execution time, sec.	$95901.678 \cdot 10^{-5}$	$3.003 \cdot 10^{-5}$	$4.004 \cdot 10^{-5}$

4 Заключение

Планирование – одна из наиболее широко изучаемых областей операционных исследований, что во многом связано с большим разнообразием различных типов задач в этой области. В данной статье был представлен анализ одной из таких задач – задачи об оптимальном расписании с идентичными исполнителями. Доказана **NP**-полнота поставленной задачи, рассмотрены различные алгоритмы её аппроксимации, доказана верхняя оценка на коэффициент аппроксимации одного из таких алгоритмов. Проведён сравнительный анализ времени выполнения различных алгоритмов аппроксимации, а также их коэффициентов аппроксимации, на случайной сгенерированной выборке экземпляров поставленной задачи.

Ссылки

[1] T.C.E. Cheng, C.C.S. Sin, The NP-completeness of the n/m/parallel/Cmax preemptive due-date scheduling problem, Mathematical and Computer Modelling, Volume 13, Issue 3, 1990, pp. 93-94.

[2] M. R. Garey and D. S. Johnson, Computers and Intractability: a Guide to the Theory of NP-completeness. Freeman, New York, 1979.

[3] Xiao Xin. A Direct Proof of the 4/3 Bound of LPT Scheduling Rule. FMSMT, 2017.