Задачи теории расписаний

Григорьева Наталья Сергеевна

СПбГУ

2024

План доклада

- Что такое задача составления расписания?
- 1. Задачи составления расписаний с учетом затрат на передачу данных
- 2. Задачи, в которых задано количество процессоров, необходимых для выполнения задания.
- 3. Он лайн расписания. Две основные модели он-лайн расписаний

Введение

Задачи упорядочения носят самый общий характер. Они возникают там, где существует возможность выбора той или иной очередности выполнения работ: при распределении работ на производстве, составлении расписания приземления самолетов, составлении расписания движения поездов, обслуживании клиентов в обслуживающих системах, управления проектами, управлении персоналом и т.д. Результаты ТР применяются при создании аппаратно-программных комплексов параллельной обработки данных и при усовершенствовании микроархитектуры процессоров.

Основные определения

Будем говорить, что составить расписание значит найти для каждой работы (операции) i время начала выполнения этой работы (операции) τ_i и указать процессор, на котором она должна выполняться.

Если допустимы прерывания, то необходимо задать время начала каждого интервала выполнения задания, суммарная длина всех интервалов, в которых выполняется задание i, должна быть равна t_i.

В каждый момент времени процессор может выполнять только одно задание.

Расписание называется допустимым, если выполняются все наложенные ограничения.

Постановка задачи с учетом затрат на передачу данных

Любая подобная программа в общем случае представляет собой ориентированный ациклический граф (DAG), дуги и вершины которого имеют определенный вес. Каждая вершина представляет собой задание, а ее вес — время выполнения данного задания.
Каждая дуга определяет отношение предшествования между двумя заданиями, а ее вес — временную затрату на передачу данных между процессорами, на которых расположены эти задания.
В случае, если задания размещаются на один и тот же процессор, то считается, что время передачи данных между ними равно нулю.

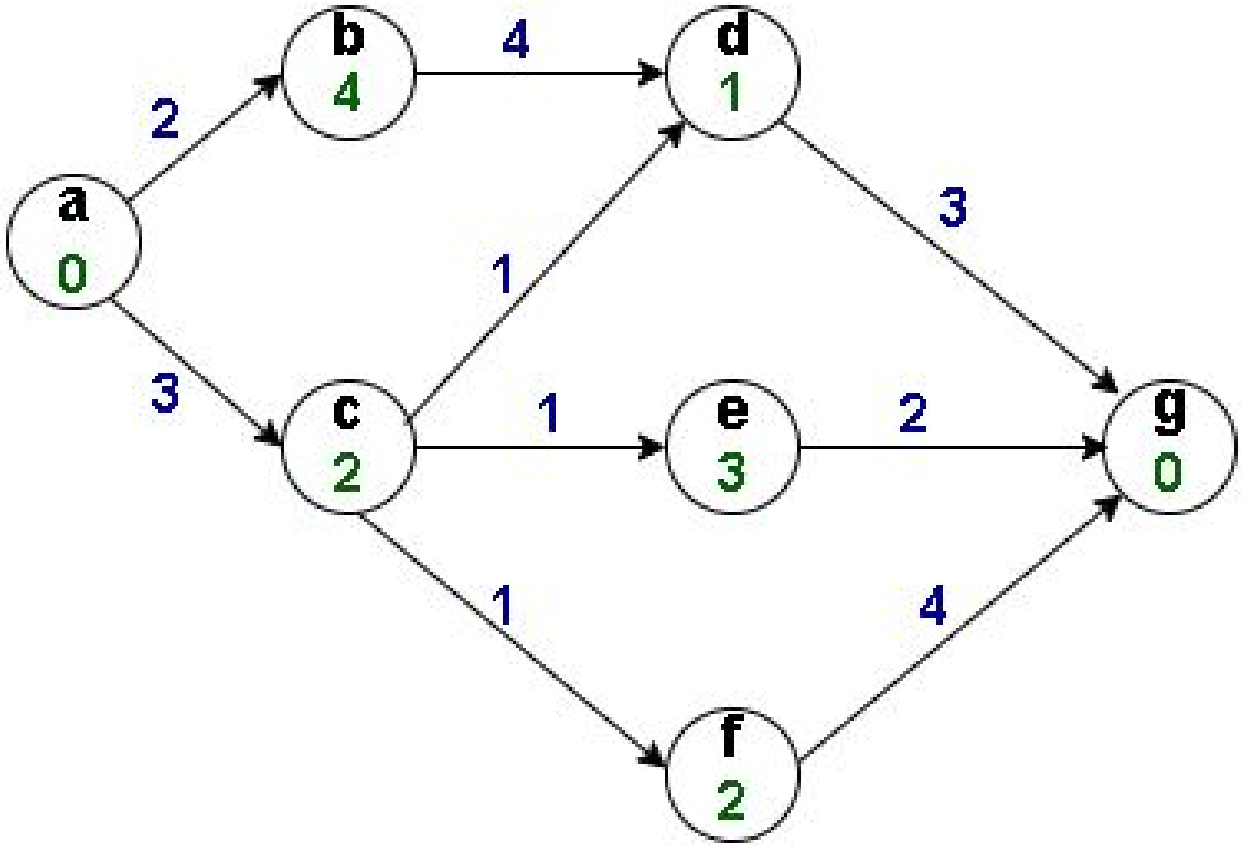
Обозначения

Граф $G(V, E, C, T)$, где

- V - множество заданий (вершины графа)
- E - множество дуг, определяющие зависимость между заданиями
- C - затраты на передачу данных (стоимости дуг)
- T - время выполнения заданий (значения укажем в вершинах графа)

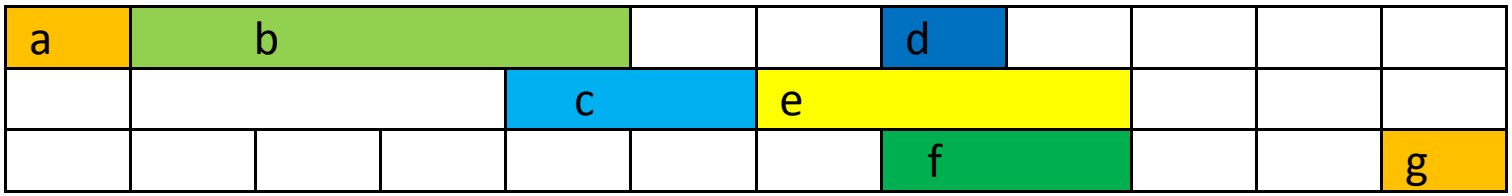
Пример

Пусть имеется 3 процессора и 7 работ.



Расимсание, построеннок алгоритмом MCP

Второе расписание генерируется, если расрешено дублирование заданий, мы видим. что дублирование задания a приводит к сокращениею общего времени выполнения.



Задачи составления расписаний с учетом затрат на передачу данных

Одной из основных проблем эффективного выполнения программы на мультипроцессорных системах является разделение программы на подзадачи, которые могут быть назначены на различные процессоры системы для параллельного выполнения.

Если такое разделение имеет высокую степень параллелизма, то требуются более значительные затраты на передачу данных между процессорами. С другой стороны, если затраты на передачу данных ограничить, то возможность в значительной степени распараллелить программу будет потеряна.

Таким образом основной целью всех алгоритмов является разделение программы на подзадачи такого размера и количества, чтобы достичь компромисса между издержками на передачу данных и степенью параллелизма и при этом добиться минимального времени завершения выполнения программы.

Классификация алгоритмов

Списочные алгоритмы - алгоритмы, определяющие классический подход к решению задачи теории расписаний — они приписывают приоритеты заданиям и затем ставят задания на процессоры на выполнение согласно списку приоритетов.
Алгоритмы декомпозиции графа - алгоритмы, базирующиеся на разбиении графов.

Постановка задачи

Любая подобная программа в общем случае представляет собой ориентированный ациклический граф (DAG), дуги и вершины которого имеют определенный вес. Каждая вершина представляет собой задание, а ее вес — время выполнения данного задания.
Каждая дуга определяет отношение предшествования между двумя заданиями, а ее вес — временную затрату на передачу данных между процессорами, на которых расположены эти задания.
В случае, если задания размещаются на один и тот же процессор, то считается, что время передачи данных между ними равно нулю.

Критические алгоритмы

Алгоритмы могут быть классифицированы на статические и динамические:
Статическими называются алгоритмы,внутри которых приоритеты присваиваются заданиям до начала процесса постановки заданий на процессоры.
Динамическими называются алгоритмы, внутри которых приоритеты заданий определяются динамически во время процесса постановки на процессоры. Приоритеты заданий пересчитываются после того, как какое-либо задание ставится на процессор, чтобы захватить изменения в относительной важности заданий.

Алгоритмы

Алгоритм обнуления дуг
Алгоритм "ранний-первый" ETF
Модифицированный алгоритм критического пути MCP
Алгоритм динамического критического пути (DCP)
Алгоритм вставки (Insertion Scheduling Heuristic) ISH
Алгоритм динамических уровней (Dynamic Level Scheduling) DLS
Алгоритм домини рующей последовательности DSC

Коэффициент зернистости графа заданий

Определим коэффициент зернистости графа заданий, учитывая затраты на передачу данных, следующим образом:

$$Granularity = \frac{1}{N-S}\left(\sum_{i=1}^{N-S}\frac{w_i}{max[w_{e_{ij}}]}\right)$$

где N — количество вершин в графе, S — количество конечных вершин в графе, w_i — вес i-ой вершины в графе, которое не является конечным, $max[w_{e_{ij}}]$ — максимальный вес дуги, идущей из вершины i в какую либо вершину j.
Коэффициент зернистости равный 1 показывает уравновешенность между затратами на передачу данных и временами выполнения заданий. Если коэффициент меньше 1, то более эффективные расписания могут быть получены, если объединять задания в более крупные кластеры, чтобы было задействовано меньше процессоров и тем самым меньше времени тратилось бы на передачу данных. Когда коэффициент зернистости велик, то нужно увеличивать степень параллелизма программы.

Сводная таблица

Были проведены вычислительные эксперименты с большим количестов тестов, которые иллюстрируют слабость критических алгоритмов. Объединяя результаты для различных классов графов , получаем следующую сводную таблицу:

Таблица 6.

	ISH	DCP	DSC	MCP	DLS	ETF
G<0.08	46.6	49.7	64.2	52.7	51.6	96.0
0.08<G<0.2	23.8	16.9	33.3	25.9	32.5	88.6
0.2<G<0.8	1.9	1.2	1.0	1.6	3.0	51.9
0.8<G<2.0	0.4	0.2	0.0	0.6	1.1	15.2
2.0<G	0.0	0.0	0.0	0.0	0.0	1.3

Таблица 6 показывает долю графов каждого из пяти вышеопределенных классов, для которых коэффициент ускорения превышает 1, т.е. лучше все задания поставить на один процессоп, чем использовать расписание, сгенерированное алгоритмом. Легко заметить, что для графов с небольшим коэффициентом зернистости "критические" алгоритмы показывают довольно плохие результаты.

Задача, в которой у задания есть ширина

Рассмотрим следующую задачу построения оптимального расписания: пусть заданы

- m **одинаковых** процессоров,
- n **независимых** задач ($j = 1, \ldots, n$),
- для каждой задачи j определено её время выполнения $p_j \in \mathbb{N}$ (длина задачи),
- кроме того, каждая задача должна выполняться ровно на $m_j \leqslant m$ процессорах (ширина задачи),
- наконец, для каждой задачи задано её время поступления $r_j \in \{0\} \cup \mathbb{N}$.

В качестве целевой функции рассматривается длина расписания C_{max}. При этом, рассматриваются как задачи с прерываниями $\langle P|p_j, r_j, m_j, pmtn|C_{max}\rangle$, так и без: $\langle P|p_j, r_j, m_j|C_{max}\rangle$.

Жадный алгоритм

Для задачи $\langle P|p_j, r_j, m_j|C_{max}\rangle$ списочный алгоритм работает следующим образом: в каждой **контрольной точке** (поступление новой задачи или завершение активной) все готовые задачи сортируются по невозрастанию в соответствии с некоторыми приоритетами и распределяются по свободным процессорам "жадным образом" (если задача слишком широка и не помещается на совбодные процессоры, она пропускается). m_j-списочным алгоритмом называется списочный алгоритм, в котором приоритеты выбираются равными ширине задачи m_j. Тем самым в каждой контрольной точке дослупные задачи сортируются по невозрастанию m_j.

Гарантированные оценки точности

Для задачи $\langle P|p_j, r_j, m_j, pmtn|C_{max}\rangle$, m_j-списочный алгоритм с прерываниями генерирует расписание с C_{max} не более чем в $2 - \frac{1}{m}$ раз хуже оптимльного.
Для задач $\langle P|p_j, r_j, m_j|C_{max}\rangle$ и $\langle P|p_j, r_j, m_j, pmtn|C_{max}\rangle$, списочный алгоритм без прерываний с любыми приоритетами генерирует расписание с C_{max} не более чем в 2 раза хуже оптимльного.

Эвристические алгоритмы

Для задач $\langle P|p_j, r_j, m_j|C_{max}\rangle$ и $\langle P|p_j, r_j, m_j, pmtn|C_{max}\rangle$, списочный алгоритм без прерываний с любыми приоритетами генерирует расписание с C_{max} не более чем в 2 раза хуже оптимльного.
Оптимальное расписание для $\langle P|p_j, r_j, m_j|C_{max}\rangle$ не более чем в два раза длиннее оптимального расписания для $\langle P|p_j, r_j, m_j, pmtn|C_{max}\rangle$.

Он-лайн задачи

Конец первой части. Он-лайн задачи рассмотрены во второй презентации.

Литература

- Graham, R.L., Lawner, E.L., Rinnoy Kan,A.H.G.: Optimization and approximation in deterministic sequencing and scheduling. A survey. Ann. of Disc. Math. **5** (10), 287–326 (1979)
- Brucker Scheduling Algoritms. (fifth edition) Springer, New York 2007
- Pinedo M. Scheduling: Theory, Algorithms, and Systems. 2014

Задача составления он-лайн расписания

Григорьева Наталья Сергеевна

Санкт-Петербург, 2024

Постановка задачи минимизации среднего времени завершения

Задача он-лайн расписания - это задача составления расписания, для которой заранее не известно время поступления задачи. В этом докладе будут рассмотрены следующие задачи:

- задача минимизации среднего времени завершения на идентичных параллельных машинах;
- модели с прерываний и без прерываний;
- задача минимизации общего времени завершения.

Постановка задачи

Рассмотрим задачу составления расписания заданий, поступающих в течение долгого времени на идентичные параллельные машины. Каждая из m машин може обрабатывать только одно из n заданий одновременно. Каждая задача

u_j имеет положительное время выполнения $t(u_j)$ и неотрицательный вес $\omega(u_j)$.
Он-лайн расписанием будет называться такое расписание, для которого характеристики задачи становятся известны, только тогда, когда задача поступит.
Составить такое расписание - определить для каждого задания u_j время начала задания и номер процессора, на котором это задание будет выполнено.

Постановка задачи

Если C_j означает время завершения задачи u_j в допустимом расписании, то соответствующее значение целевой функции -

$$\sum_{i=1}^{n} \omega(u_j) C_j$$

Будем рассматривать ситуации с прерыванием и без прерывания. В случае модели с прерываниями, обработка задачи может быть приостановлена и возобновлена позже на любой машине без затрат.

Расписание для m идентичных параллельных процессоров

Задача оф-лайн расписания с одинаковыми временами поступления оптимально решается с помощью правила кратчайшего времени обработки Смита (WSPT), которое располагает задачи в невозрастающем порядке отношения веса к времени выполнения.
Для удобства, допустим, что задачи проиндексированы в таком следующем порядке: $\frac{\omega(u_1)}{t(u_1)} \geq \cdots \geq \frac{\omega(u_n)}{t(u_n)}$.
Качество он-лайн расписания обычно оценивается следующим соотношением. Пусть s - расписание, f_A - значение целевой функции для алгоритма A ,f_{opt} значение целевой функции для оптимального off лайн расписания, тогда если

$$\frac{f_A}{f_{opt}} \leq z$$

будем говорить, что алгоритм имеет гарантированную оценку точности равную z и называть алгоритм z-конкурентоспособным.

Расписание для m идентичных параллельных процессоров

Идея состоит в том, чтобы прерывать активную задачу с меньшим приоритетом, как только новая, более приоритетная задача появляется, и нет свободного процессора в этот момент.
В случае без прерывания, на освободившийся процессор назначается наиболее приоритетная задача из доступных.
В этом случае неудачным назначением является установка на процессор длинной задачи с низю ким приоритетом, а задача с высшим приоритетом выполняется, после начала длинной менее приоритетной задачи.
Предлагается изменить время поступления каждой работы, так, чтобы время поступления задачи было согласовано с временем выполнения задачи.
Ниже мы рассмотрим семейство подобных алгоритмов и покажем, что лучший результат достигается при $z = 3.28$.

Расписание для параллельных машин с прерыванием

Рассмотрим расширение WSPT правила для m идентичных параллельных процессоров. В каждый момент времени процессоры выполняют m работ с высшими приоритетами из доступных. Мы можем прерывать выполнение активной в настоящее время работы, если необходимо.

Алгоритм P-WSPT

Алгоритм работает в режиме реального времени, решение принимается либо в момент освобождения какого-либо процессора, либо в момент поступления нового задания.

Решение о том какую работу запустить в каждый заданный момент времени t зависит только от приоритетов доступных работ.

Теорема 1. Для алгоритма P-WSPT верно

$$\frac{f_A}{f_{opt}} \leq 2$$

Расписание для параллельных машин с прерыванием

Этапы алгоритма:

- Определим набор новых заданий.
- Найдем задание с максимальным приоритетом. $\frac{(u_o)}{t(u_0)} = max\frac{(u_i)}{t(u_i)}, u_i \in \mathbf{N}$ Если готовых заданий нет, то простой процессоров до поступления новых заданий.
- Назначение наиболее приоритетного задания на свободный процессор.
- Если все процессоров заняты, то найдем наименее приоритетное задание, из тех, что стоят на процессорах, прервем его и поставим задание u_0
- Если вновь поступившее задание не является приоритетным, ожидание освобождения какого-либо процессора.
- Если все поступившие задачи выполнены - вернуться к шагу 1.
- Выход из цикла, как только все задания будут выполнены.

Расписание для параллельных машин с прерыванием

Семейство примеров показывает что результат **Теоремы 1** не может быть улучшен.

Лемма 1. Значение конкурентоспособности алгоритма P-WSPT не лучше, чем 2 для проблем с неполной информацией для любого количества машин.

Следующий пример показывает, что оценка конкурентносроспособности P-WSPT расписания стремиться к 2, когда n стремится к бесконечности.

Расписание для параллельных машин с прерыванием

Рассмотрим пример. Пусть есть $n+1$ задание, для которых $w(u_j) = 1; t(u_j) = n - j/n; r(u_j) = jn - j(j+1)/2n, 0 \leq j \leq n$. Имеется m копий этих заданий.

Алгоритм P-WSPT:

1 пр.	0	1	2	…	n	n-1	…	0
…	…	…	…	…	…	…	…	…
m пр.	0	1	2	…	n	n-1	…	0
C_j [1]	$r_1 \mid прер. \mid$	$r_2 \mid прер. \mid$	$r_3 \mid прер. \mid$	…	$r_n + p_n$	$r_n + p_n + \frac{1}{n}$	…	$r_n + p_n + \frac{n}{n}$

Расписание для параллельных машин без прерывания

Каждый алгоритм для он-лайн задаче должен использовать что-то вроде ждущей стратегии.
Изменим время поступления работы j на $r^*(u_j)$, для которого верно:

$$max\{r(u_j), \alpha(u_j)p(u_j)\} \leq r^*(u_j) \leq r(u_j) + \alpha(u_j)p(u_j),$$

для некоторого $\alpha \in (0, 1]$. Всякий раз, когда машина освобождается, выбираем среди доступных задач задачу j с наивысшим приоритетом и запускаем её на незанятой машине.

Алгоритм "Исправленный"WSPT:

- Расчёт верхней и нижней границы для $r^*(u_j)$.
- Определение значения $r^*(u_j)$ из полученного интервала
- Сортировка заданий по временам поступления $r^*(u_j)$.
- Поиск новых заданий
- Рассчитаем приоритет вновь поступивших заданий, если заданий нет, то простой процессоров до поступления новых заданий, т.е. возврат к шагу 4.
- Назначение наиболее приоритетного задания на свободный процессор. Если все процессоров заняты, ожидание освобождения какого-либо процессора.
- Если все поступившие задачи выполнены, то возврат к шагу 4.
- Выход из цикла, как только все заданий будут выполнены.

Расписание для параллельных машин без прерывания

Идея исправленного времени поступления гарантирует, что время выполнения любой задачи не слишком велико по сравнению со временем её поступления. Проанализируем выполнение "исправленного"WSPT алгоритма.

Теорема 2. "исправленный"Алгоритм WSPT принимает конкурирующее ограничение менее, чем $2 + max\{\frac{1}{\alpha}, \alpha + \frac{m-1}{2m}\}$, для задачи 1.

Несложные вычисления показывают, что минимум $\max\{\alpha, \alpha + (m-1)/(2m)\}$ достигается в $\alpha = (1 - m + sqrt(16m2 + (m - l)2))/(4m)$ положим $\alpha_m := \alpha$. В частности, $\alpha_1 = 1$.

Следствие 1. "Исправленный"алгоритм WSPT $\alpha = \alpha_m$ является $(2 + \frac{1}{\alpha_m})$-конкурентоспособным. Значение этой возрастающей функции от m имеет предел $\frac{9+\sqrt{17}}{4} \approx 3.28$ при $m \mapsto \infty$. Для одномашинного случая алгоритм является 3- конкурентноспособным..

Расписание для параллельных машин без прерывания

Лемма 3. "Исправленый"алгоритм WSPT не может достигать лучших конкурентных отношений, чем $\max\{\frac{1}{\alpha}, \alpha + \frac{m-1}{2m}\} \leq 2 + \frac{\sqrt{5}-1}{2}$, при $\alpha \in (0, 1]$ для любого количества машин.

Постановка задачи минимизации общего времени завершения

Рассматривается задача составления расписания на m параллельных идентичных процессорах, где задачи поступают с течением времени. Задачи независимы друг от друга, их количество и необходимое время выполнения заранее неизвестно.
Требуется построить расписание с минимальным временем завершения всех задач.
Для построения расписания применяется алгоритм LPT (longest processing time).

LPT-алгоритм.

Алгоритм LPT описывается следующим образом:
Каждый раз, когда освобождается один из процессоров, на него ставится доступная задача с наибольшим временем выполнения, если доступных задач нет, ожидается следующая.

Пусть r_j, p_j – время поступления и выполнения задачи J_j соответственно. Для расписания σ положим $C_{max}(\sigma)$ – время завершения расписания, $S_j(\sigma), C_j(\sigma)$ – время начала и завершения выполнения задачи J_j в расписании.

Основная теорема.

LPT-алгоритм имеет гарантию производительности $\frac{3}{2}$, при этом граница жёсткая.

Оценка снизу.

Следующая теорема показывает, что данный результат вполне неплох относительно других онлайн алгоритмов.
Любой онлайн алгоритм имеет коэффициент производительности в худшем случае не менее $1 + \alpha \approx 1.3473$, где α есть решение уравнения $\alpha^3 - 3\alpha + 1 = 0$ в интервале $\left[\frac{1}{3}, \frac{1}{2}\right]$.
Для $m = 2$, граница может быть улучшена до $(5 - \sqrt{5})/2 \approx 1.382$.

Спасибо за внимание!

Литература


Graham, R.L., Lawner, E.L., Rinnoy Kan,A.H.G.: Optimization and approximation in deterministic sequencing and scheduling. A survey. Ann. of Disc. Math. **5** (10), 287–326 (1979)


Brucker Scheduling Algoritms. (fifth edition) Springer, New York 2007


Pinedo M. Scheduling: Theory, Algorithms, and Systems. 2014