**Планирование машины пакетной обработки с учетом ограничений приоритета, дат выпуска и одинакового времени обработки**

**T.C.E. ,** и

1 Департамент управления, Гонконгский политехнический университет, Хунг Хом, Коулун, Гонконг, Китайская Народная Республика

2 Математический факультет, Университет Чжэнчжоу, Чжэнчжоу, Хэнань 450052, Китайская Народная Республика

**АННОТАЦИЯ**

Для минимизации регулярной целевой функции мы рассматриваем задачу планирования параллельного пакетирования для одной машины с отношениями предшествования, датами выпуска и с одинаковым временем обработки. Когда время обработки составляет единицу, мы полагаем время выполнения оптимального алгоритма . Когда нет предшествующих отношений, мы решаем эту проблему путем динамического программирования за . времени. Когда отношения предшествования «слоисто завершены», мы решаем эту проблему с помощью алгоритма динамического программирования, который выполняется за . времени. Для задачи минимизации полного взвешенного времени завершения мы полагаем время для алгоритма аппроксимации. Для задачи минимизации рабочего времени мы полагаем время для оптимального алгоритма и выражение минимального рабочего времени.

Ключевые слова: планирование, пакетная обработка, ограничения приоритетов, даты выпуска, одинаковое время обработки, алгоритмы.

**1 Введение и постановка задачи**

Пусть заданы *n* заданий , ,..., и один компьютер, который может одновременно обрабатывать пакетные задания. Существуют отношения приоритета ≺ между заданиями. Каждое задание имеет целочисленное время обработки и целочисленную дату выпуска . Задания обрабатываются пакетами, где пакет является подмножеством заданий, и мы требуем, чтобы эти пакеты осуществили раздел множества всех заданий. Время обработки пакета равно наибольшему времени обработки всех заданий в пакете.

Предположим, что задана пакетная последовательность (которая указывает порядок обработки определенного пакетного раздела заданий), и мы будем обрабатывать пакетные задания в соответствии с этой пакетной последовательностью. Мы требуем, чтобы время запуска пакета было как минимум максимальной датой выпуска заданий в нем, и это максимальное значение можно рассматривать как дату выпуска пакета. Если и являются двумя заданиями, такими, что , мы также требуем, чтобы обрабатывался после времени завершения ; поэтому и не могут быть обработаны в одном пакете. Время завершения всех заданий в пакете определяется как время завершения пакета. Следуя [1] и [7], мы назовем эту модель проблемой планирования параллельного пакетирования и обозначим ее как

,

где - это целевая функция, которая является функцией времени завершения задания по заданному расписанию, и которая должна быть минимизирована. В этой статье мы будем предполагать, что целевая функция регулярна [1], т. е. неубывает в .

Для задачи , возможное расписание задается последовательностью

такой, что для любой пары заданий и с , если и , то *x < y*. Поскольку целевая функция является регулярной, мы предполагаем, что все задания в одном и том же пакете запускаются одновременно в самое ближайшее время запуска. Следовательно, время начала каждого пакета определяется последовательностью пакетов. Для каждого пакета , если время начала равно , тогда время завершения просто

.

Проблема параллельного пакетного планирования является одной из важных современных моделей планирования, которой уделяется большое внимание в литературе. Фундаментальная модель задачи параллельного пакетного планирования была впервые представлена Ли и др. в [8] с ограничением на то, что количество заданий в каждом пакете ограничено числом *b*, которое обозначается . Эта ограниченная модель мотивируется операциями обжига при производстве полупроводников [8]. Например, партия интегральных микросхем (заданий) помещается в печь ограниченного размера, чтобы проверить их термостойкость. Микросхемы нагреваются внутри печи до тех пор, пока не сгорят все. Время выгорания микросхем (время обработки задания) может быть разным. Когда микросхема сгорела, она должна ждать внутри печи, пока не сгорят все микросхемы. Следовательно, время обработки партии микросхем - это время обработки самого длинного задания в партии.

Подробное обсуждение неограниченной версии исследуемой проблемы приведено в [2]. Эта неограниченная модель может быть применена, например, к ситуациям, когда содержимое партии необходимо закалить в достаточно большой печи, и поэтому размер партии не ограничен [2].

Последние разработки по этой теме можно найти в книге [1] и на сайте [3]. Кроме того, в [4], [6], [9] и [10] представлены новые результаты сложности по проблеме параллельного пакетного планирования с учетом дат выпуска. Мы будем рассматривать только неограниченную версию задачи параллельного пакетного планирования.

Для задачи в [1] подразумевается, что для каждой регулярной целевой функции существует алгоритм времени . Для задачи , Чэн и др. [5] недавно показали, что даже самые простые задачи и сильно NP-трудны.

Мы рассматриваем задачу . В [3] сообщается, что сложность задачи неизвестна даже для таких обычных регулярных целевых функций, как время выполнения; максимальная задержка ; общее время завершения ; и полное опоздание , где - срок оплаты , а .

В этой статье мы покажем, что задачу планирования можно решить за время, можно решенить за время, а можно решить за время. Мы даем время алгоритм аппроксимации для задачи . Также показано, что задачу можно решить за время. Кроме того, мы даем выражение минимального рабочего времени.

**2 Изменение даты выпуска**

Рассмотрим задачу , где *f* - любая регулярная целевая функция. Эта проблема может быть решена с помощью простого алгоритма.

Мы предполагаем, что приведенное здесь перечисление заданий является топологическим, то есть для любых двух заданий и с , мы должны иметь *i <j*. Согласно Брукеру [1], перечисление топологических заданий может быть получено за время по стандартному «Алгоритму топологического перечисления».

Если и являются двумя заданиями, такими, что , то время запуска пакета, содержащего , должно быть не меньше, чем в любом возможном расписании. Установив изменение даты выпуска каждого задания

,

мы видим, что значение целевой функции не изменится ни при каком возможном расписании. Следовательно, мы можем рекурсивно изменять даты выпуска заданий так, чтобы для каждой пары заданий и , если , тогда .Эта процедура может быть выполнена за время стандартным «алгоритмом модификации » в Брукере [1].

Важным наблюдением является то, что при измененных датах выпуска ., если . Кроме того, в любом возможном расписании время запуска пакета, содержащего , составляет по меньшей мере . Это значительно упрощает описание алгоритмов, обсуждаемых в этой статье. Следовательно, в оставшейся части статьи мы предполагаем, что даты выпуска изначально были изменены таким образом, что для каждой пары заданий и , если , то .

Рассмотрим специальный случай, когда существует некоторое целое число *e* c *0 ≤ e ≤ p - 1*, такое, что для каждого существует такое , что , *1 ≤ j ≤ n*. Предположим, у нас есть *k* различных дат выпуска таких, что для *1 ≤ i ≤ k - 1*. Отметим, что означает, что , мы можем сформировать пакетную последовательность

за время путем установки

, для *1 ≤ x ≤ k.*

Ясно, что является возможным расписанием, так что время начала и время завершения каждого пакета равны и соответственно. Поскольку любое задание начинается в дату его выпуска (т.е. самое раннее возможное время запуска) , то пакетная последовательность *BS* должна быть оптимальной для любой обычной целевой функции *f*.

Исходя из вышеизложенного, проблему можно решить по следующему правилу пакетирования.

**Алгоритм 2.1 Правило пакетирования для системы заданий с .**

В каждой точке сформируйте следующий первый пакет, включив в него все доступные непакетированные задания, которые не имеют непакетированных предшественников.

Когда для всех заданий , *e = 0*, то указанное выше правило пакетирования решает проблему .

**3 Система заданий при одинаковом времени обработки**

Определим

*.*

Понятно, что должно существовать оптимальное расписание для такое, что время начала и время завершения каждого пакета принадлежат T. Следовательно, достаточно рассмотреть расписание с временем запуска пакета в T.

Учитывая экземпляр проблемы , пусть . Мы определяем слои заданий следующим образом:

,

.

называется *i*-й слой системы заданий. Пусть . Тогда и каждый слой состоит из независимых заданий. Понятно, что слои заданий можно получить за время.

Если для *1 ≤ i ≤ m - 1* каждое задание в является предшественником каждого задания в , отношение предшествования считается «слоисто завершенным». Соответствующая задача обозначается через .

**3.1 Независимая система заданий**

Специальная подзадача равна , где задания независимы (без ограничений приоритета между заданиями). Сначала рассмотрим проблему

при ограничении, что время начала первой партии составляет не менее *s*, а время завершения последней партии - не более *t*, где либо , либо .

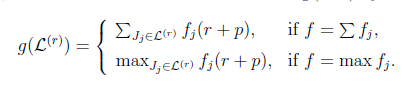
Пусть обозначает оптимальное (минимальное) объективное значение для . Мы принимаем соглашение, что , если *L = ∅*, и *= + ∞,* если не имеет выполнимого расписания. Пусть

.

Тогда . Ясно, что существует оптимальное расписание для , такое, что время начала первой партии принадлежит . Предположим, что время начала первой партии *r*, и пусть



Так как *f* регулярна, сформирует первую партию. Определим как



Ясно, что для вычисления требуется не более *O(|L|)* времени. Для ,мы имеем следующую рекурсию динамического программирования



Для мы имеем



В процедуре динамического программирования нижние границы времени начала подмножеств заданий *L* выбираются из



и у нас есть таких выборов. Когда новая нижняя граница времени запуска *r* установлена, новое подмножество заданий формируется однозначно. Каждая рекурсия выполняется за *O(|L|)*. Следовательно, когда заданы *s* и *t*, обе рекурсии динамического программирования выполняются за времени, чтобы решить проблему .

Для общей задачи , множество заданий *L = J*. Так как



можно сделать заключение, что задача может быть решена за время.

**3.2 Слоисто завершенная система заданий с ограниченными приоритетами**

Теперь обратим внимание на проблему . Пусть , , ..., - слои заданий в ≺. Поскольку каждое задание в должно завершить свою обработку перед началом любого задания в ,то это позволяет нам использовать динамическое программирование для решения этой проблемы. Пусть



Для мы рассматриваем проблему

**

с ограничением, что время начала первой партии составляет не менее *s*. Пусть обозначает оптимальное (минимальное) объективное значение для . Мы принимаем соглашение, что и . Пусть



Тогда у нас есть динамическое программирование рекурсий



для , и



для . Теперь у нас есть *m* вариантов для *i*, вариантов для *s ∈ T* и варианта для *t ∈ R(i, s)*. Перед рекурсией для вычисления всех для всех *i*, *s* и *t* требуется времени. Следовательно, сложность каждой из двух рекурсий динамического программирования составляет . Так как оптимальное значение задачи задается как , то мы заключаем, что задача планирования может быть решена за время.

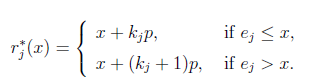
**3.3 Алгоритм аппроксимации**

Хотя сложность остается открытой, мы не ожидаем простого алгоритма за полиномиальное время для этой задачи. Ниже мы приведем алгоритм округления полиномиального времени по дате выпуска.

Обозначим каждую дату выпуска как



Пусть *x* будет целым числом с *0 ≤ x ≤ p - 1*. Даты выпуска округляются следующим образом:



Важным наблюдением является то, что , для *1 ≤ j ≤ n.*

**Лемма 3.3.1**. Если , то .

**Доказательство.**  Поскольку , имеем . По той причине, что и , получаем, что и, следовательно . Из этого следует, что можно разделить на *p*.

При датах выпуска задача может быть решена с помощью алгоритма 2.1 для получения оптимального расписания *π(x).* Этот график *π(x)* явно выполним для и поэтому может быть использован в качестве приближенного решения.

Поскольку время начала каждого задания при π(x) равно , мы имеем . Итак, объективное значение определяется

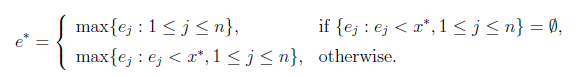
.

Cледующая лемма показывает, что, чтобы минимизировать для *0 ≤ x ≤ p - 1*, нам нужно только минимизировать для *1 ≤ j ≤ n.*

**Лемма 3.3.2** Существует целое число *j*, *1 ≤ j ≤ n* такое, что

*.*

**Доказательство**. Предположим что такое, что . Если , то мы определим

**

Можно проверить, что и, следовательно, , *1 ≤ j ≤ n*. Поскольку регулярна, мы должно иметь . Результат следует.

Теперь наш алгоритм округления даты выпуска можно обобщить следующим образом.

**Алгоритм 3.3.3. Округление даты выпуска.**

1. Выберете так, чтобы

.

1. Примените Алгоритм 2.1 к задаче планирования , чтобы получить расписание .

Алгоритм 3.3.3 является полиномиальным и имеет хорошую производительность для и . Для , абсолютная ошибка может быть ограничена сверху по , так как для *1 ≤ j ≤ n*. Далее мы оценим верхнюю границу коэффициента производительности алгоритма 3.3.3 для .

**Теорема 3.3.4** Алгоритм 3.3.3 является алгоритмом 3/2-аппроксимации полиномиального времени.

Доказательство. Пусть - оптимальное решение для . Тогда, . Поскольку , имеем

**Пункт 1 **

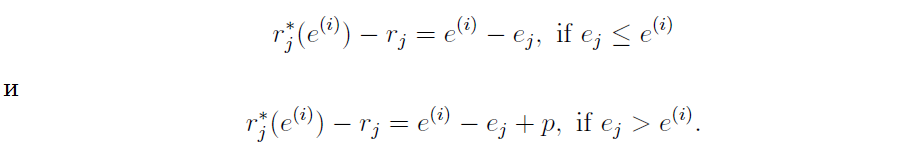
Предположим, что и пусть

будет таким, что . Запишем 

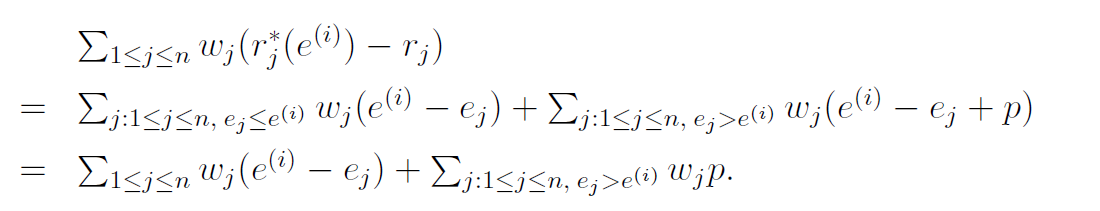
Поскольку  и  мы дополнительно имеем Из этого следует

**Пункт 2** 

По определению для и , мы имеем

**

Следовательно, для мы имеем



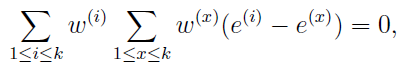
Это может быть переписано как

**Пункт 3** для .

Пусть Тогда по п.2 и п.3 мы имеем

**Пункт 4** 

Отмечая тот факт, что



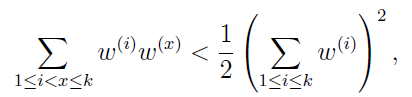
мы можем легкий вывод о том, что



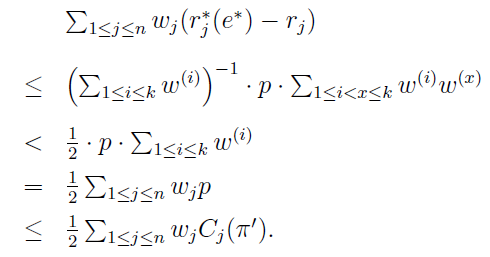
Эквивалентно, мы имеем

**Пункт 5** 

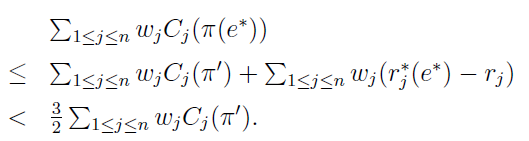
Поскольку



Мы выводим из п.4 и п.5, что



Из пункта 1 следует, что



Результат следует.

Мы предполагаем, что оценка 3/2 в теореме 3.4 может быть дополнительно улучшена.

**4 Минимизация рабочего времени и расширение**

**4.1 Минимизация рабочего времени**

Задача , обозначенная в дальнейшем буквой *P*, может быть легко решена с помощью следующего алгоритма.

**Алгоритм 4.1.1. Правило дозирования при минимизации рабочего времени**

В каждой точке сформируйте следующий последний пакет, включив в него все непакетированные задания, у которых нет неразмеченных преемников.

Ясно, что сложность алгоритма 4.1 равна . Корректность алгоритма 4.1 вытекает из следующей теоремы.

**Теорема 4.1.2** Рабочее время для *P*, полученное по алгоритму 4.1 составляет .

**Доказательство**. Предположим, что это пакетная последовательность, полученная с помощью алгоритма 4.1. Пусть . Тогда . Для каждой пары смежных пакетов и , , пусть таково, что . По правилу пакетирования алгоритма 4.1 должен существовать некоторый такой, что . Тогда . Это означает, что , для . Следовательно, при пакетной последовательности *BS*, каждый пакет имеет время начала и время завершения . Результат следует.

**4.2 Расширение**

Мы рассматриваем расширение проблемы *P*. На практике фирмы стремятся снизить стоимость хранения для заказанных товаров. Таким образом, они сталкиваются с проблемой планирования первичного-вторичного критерия. Первый критерий - минимизировать , а второй критерий - максимизировать . Обозначим эту первично-вторичную критериальную задачу через .

Пусть  - пакетная последовательность, полученная по алгоритму 4.1. Пусть - минимально рабочее время для *P*. По построению *BS* для каждого задания должна существовать цепочка заданий

такой, что для *i + 1 ≤ x ≤ k*. Отсюда следует, что каждое задание в имеет время завершения не более и время начала не более в любом возможном расписании. Чтобы максимизировать , мы определим как расписание, полученное из BS, так что каждый пакет имеет время начала и время завершения. Ясно, что является выполнимым расписанием, и каждое задание завершается в самое большое возможное время завершения при ограничении, что рабочее время минимизируется. Отсюда заключаем следующую теорему.

**Теорема 4.2.1.** - оптимальный график для задачи первичного-вторичного критерия .

**5 Заключение**

Параллельно-пакетная задача планирования была изучена в этой статье. Ограничение на задания с одинаковым временем обработки в значительной степени упрощает проблему, но наличие ограничений приоритета между заданиями увеличивает сложность проблемы. В этой статье мы показали, что задача можно решить за время , может быть решена за время и можно решить за . Для задачи мы дали алгоритм 3/2-аппроксимации времени . Мы также показали, что проблема может быть решена за . время. Кроме того, мы дали выражение минимального рабочего времени. Для дальнейшей работы, сложность по-прежнему открыта для регулярной целевой функции



**Благодарности**

Мы благодарны за конструктивные комментарии рецензентов по более ранней версии этой статьи. Это исследование было частично поддержано Гонконгским политехническим университетом в рамках гранта ASD в China Business Services. Последние два автора были также частично поддержаны Национальным фондом естественных наук Китая.