Материалы 41-й Гавайской международной конференции по системным наукам – 2008

**Приоритетное приближение для пакетирования**

Вольфганг Бейн ∗

Центр перспективных исследований алгоритмов

Школа компьютерных наук

Университет Невады

Лас-Вегас, Невада 89154

[bein@cs.unlv.edu](mailto:bein@cs.unlv.edu)

Джон Нога

Кафедра компьютерных наук

Калифорнийский государственный университет

Нортридж, Калифорния 91330

[jnoga@csun.edu](mailto:jnoga@csun.edu)

Джеффри Уигли

Кафедра компьютерных наук

Калифорнийский государственный университет

Нортридж, Калифорния 91330

[jeffw@csun.edu](mailto:jeffw@csun.edu)

**Аннотация**

Здесь мы рассмотрим проблему серийного пакетирования с одной машиной при средневзвешенном завершении. Эта проблема, как известно, является NP-трудной, и на сегодняшний день не существует алгоритма аппроксимации для нее. Пакетирование широко применяется в производстве, управлении решениями и планировании в информационных технологиях.

В этой статье мы даем алгоритм первого приближения для задачи. Алгоритм имеет коэффициент аппроксимации 2 и является алгоритмом приоритета, который группирует задания в порядке убывания приоритета. Мы также даем нижнюю границу для коэффициента аппроксимации любого алгоритма приоритета и предполагаем, что существует алгоритм приоритета, который соответствует этой границе. Адаптивный алгоритм экспериментов используется для поддержки гипотезы. Более простой проблемой является список версий проблемы, где указан порядок заданий. Мы даем новый линейный алгоритм времени для задачи пакетной обработки списка.

**1 Мотивация и предпосылка**

Рассмотрим *задачу пакетирования*, где набор заданий со временем обработки и весами , должен быть запланирован на одной машине, и где *J* должен быть разделен на пакеты . Все задания в одном пакете выполняются совместно, и время завершения каждого задания определяется как время завершения его пакета. Мы предполагаем, что при планировании пакета требуется время установки *s* = 1. Цель состоит в том, чтобы найти расписание, которое минимизирует сумму времен завершения , где обозначает время завершения в данном расписании. При заданной последовательности заданий алгоритм пакетирования должен назначать каждому заданию пакет. Более формально возможным решением является присвоение каждому заданию пакета .

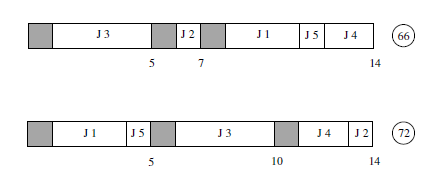


Рисунок 1 – Пример пакетирования

Например, на рисунке 1 показаны два графика для задачи из 5 заданий, где время обработки составляет = 3, = 1, = 4, = 2, = 1, а весовые коэффициенты равны = = = 1 и = = 2. (Заметим, что обведенные значения дают средневзвешенное время завершения изображенных расписаний.)

Задача, рассматриваемая в этой статье, имеет задания, выполняемые последовательно, поэтому проблема более точно называется проблемой *s-batch*. Мы отмечаем, что существует другая версия проблемы, которая здесь не изучалась, когда задания пакета выполняются параллельно, известная как проблема *p-batch*. В этом случае длина пакета - это максимальное время обработки его заданий. S-batch также обозначается в *α|β|γ* нотации как задача 1|s-batch|. Кроме того, иногда мы будем для удобства записывать данные задания в виде {(, ), (, ), … , (, )}. Таким образом, в примере мы написали бы {(3, 1), (1, 2), (4, 2), (2, 1), (1, 1)}.

Брукер и Альберс [1] показали, что проблема 1|s-batch| является NP-трудной в сильном смысле, давая сокращение от 3-PARTITION. Существует большой объем работ по проблемам пакетирования (см. [2, 3, 6, 7, 9, 12]), и пакетирование широко применяется в производстве (см. [8, 16, 20]), управлении принятием решений (см. [14]), и планировании в информационных технологиях (см. [10]). Более поздняя работа по онлайн-пакетированию связана с проблемой подтверждения TCP (Transmission Control Protocol) (см. [4, 11, 13]).

В этой статье мы даем алгоритм первого приближения для задачи 1|s-batch|. Фактически мы даем два алгоритма, один из которых называется PseudoBatch (Псевд-опакет), а другой – CanonicalBest (КаноническиЛучший). Для алгоритмов аппроксимации естественно рассматривать задания в соответствии с порядком приоритетов . Если задания пронумерованы так, что , мы говорим о *каноническом порядке*. Алгоритм, который планирует задания в этом порядке, называется *алгоритмом приоритета*. Оба наших аппроксимационных алгоритма являются приоритетными.

Напомним, что качество аппроксимации является мерой с точки зрения ее коэффициента ρ. Учитывая задачу оптимизации *P*, мы говорим, что алгоритм имеет коэффициент аппроксимации ρ, если для любого случая π ∈ *P*,

,

где это значение оптимального решения. Мы показываем, что PseudoBatch и CanonicalBest имеют коэффициент аппроксимации ρ = 2. Мы также даем нижнюю границу для коэффициента аппроксимации алгоритма любого приоритета и предполагаем, что CanonicalBest соответствует этой границе. Адаптивный алгоритм экспериментов используется для поддержки гипотезы.

Гораздо более простой вариант проблемы - это список задач, в котором указан порядок заданий, т. е. ≤, если i <j. Например, на рисунке 2 показаны три расписания для задачи с 5 заданиями {(3, 1) (1, 1) (4, 1) (2, 1) (1, 1)}. Обведенные значения показывают средневзвешенное время завершения графиков слева.

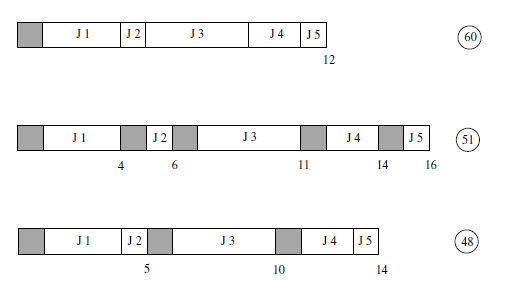


Рисунок 2 – Пакетный список

Брукер и Альберс [1] дали алгоритм линейного времени для задачи пакетной обработки списка. (Таким образом, для решения задачи 1|s-batch| достаточно знать порядок заданий в оптимальном решении.) В этой статье мы приводим альтернативный алгоритм. Наш алгоритм использует тот факт, что проблема может быть сведена к проблеме кратчайшего пути, где базовая матрица затрат является полностью монотонной матрицей и, таким образом, может использовать алгоритм поиска матриц Лармора и Шибера [15] в качестве подпрограммы. Матрица A называется *полностью монотонной*, если для всех i <i' и j <j', A [i, j]> A [i, j'] подразумевается A [i, j]> A [i', j']; Матрица A называется *матрицей Монжа*, если A [i, j] + A [i', j'] ≤ A [i', j] + A [i, j']. Очевидно, что каждая матрица Монжа является полностью монотонной. Отметим, что алгоритм пакетного списка линейного времени используется для реализации CanonicalBest за время выполнения O(*n* log *n*).

Наша статья организована следующим образом: в разделе 2 мы даем алгоритмы приоритетного приближения. Раздел 3 дает наш альтернативный алгоритм линейного времени для задачи пакетной обработки списка. В разделе 4 представлена нижняя граница коэффициента аппроксимации для любого приоритетного алгоритма. Раздел 5 описывает эксперименты с генетическим алгоритмом. В частности, мы приводим эксперимент с адаптивным алгоритмом, который поддерживает гипотезу о том, что коэффициент аппроксимации CanonicalBest соответствует нижней границе. В этом разделе также содержится описание генетического алгоритма для задачи 1|s-batch|, реализованного в GAlib, объектно-ориентированной библиотеке Мэтью Уолла [19], разработанной в MIT. Мы завершаем открытыми проблемами в разделе 6.

**2 Аппроксимационные алгоритмы**

Мы даем следующую техническую лемму, которая также известна как «Правило Смита» в области планирования.

**Лемма 1** *Даны ,…, > 0, ,…, ≥ 0 с и перестановка π. Для i = 1,…,n пусть . Тогда минимизируется, когда π - тождество.*

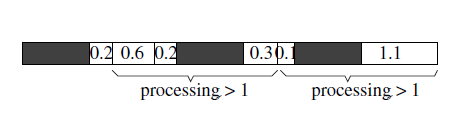
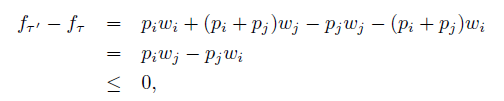


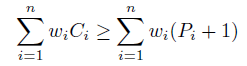
Рисунок 3 - PseudoBatch (Псевд-опакет) для = 0.2, = 0.6, = 0.2, = 0.3, = 0.1, = 1.1.

*Доказательство:* Рассмотрим перестановку τ, которая не является тождественной. Тогда τ имеет инверсию j> i , с i непосредственно перед j в τ. Пусть τ' - перестановка с заменой i и j. У нас есть



так как . Отсюда следует, что ≤ и все готово.

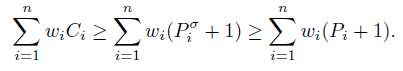
**Лемма 2** *Пусть - времена завершения оптимального расписания для задачи* 1|s-batch|*. Тогда у нас есть*



*Доказательство:* Пусть перестановка σ - порядок оптимального расписания. Тогда



По лемме 1 имеем

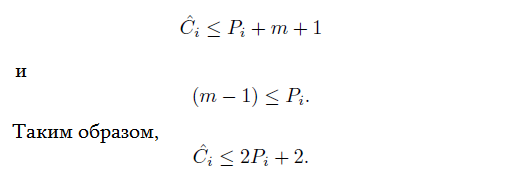


Теперь представим простой параметризованный алгоритм PseudoBatch для задачи 1|s-batch|. PseudoBatch сначала переупорядочивает задания так, чтобы они были в каноническом порядке. Затем задания назначаются пакетам в указанном порядке. После получения  наш алгоритм имеет только два варианта, а именно: назначать тому же пакету, что и , или нет. Мы используем фразу «пакеты *A* на шаге i», чтобы обозначить, что алгоритм *A* решает, что является первым заданием нового пакета, то есть = + 1. Мы используем фразу «текущий пакет», чтобы обозначить пакет, которому было назначено последнее задание. Затем, когда  получен, *A* должен решить, добавить ли  в текущий пакет или «закрыть» текущий пакет и назначить  новому пакету. PseudoBatch поддерживает переменную *P*, которая будет суммой времени обработки набора недавних заданий: мы называем этот набор текущий псевдо-пакет. Если *P*> 1, PseudoBatch пакетирует, а также устанавливает *P* в ноль. Таким образом, *i*-ый псевдо-пакет содержит все элементы, кроме первого члена *i*-го пакета, вместе с первым членом (*i* + 1)-го пакета, если только *i = r*. Каждое задание, кроме  , принадлежит только одному псевдо-пакету.

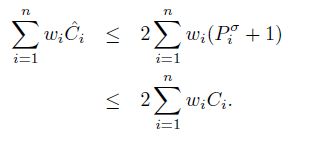
На рисунке 3 изображен пример для = 0.2, = 0.6, = 0.2, = 0.3, = 0.1, = 1.1.

**Теорема 1** PseudoBatch *имеет коэффициент аппроксимации 2.*

*Доказательство:* Как и раньше, пусть будет временем завершения оптимального расписания для задачи 1|s-batch|. Пусть обозначает время завершения заданий, когда алгоритм PseudoBatch запущен на экземпляре, и пусть *m* будет количеством пакетов, созданных алгоритмом. Очевидно, у нас есть



По лемме 2 мы имеем



Пусть CanonicalBest будет алгоритмом, который размещает задания в каноническом порядке, и будет алгоритмом линейного времени следующего раздела (или алгоритмом из [1]), чтобы получить оптимальное расписание пакетной обработки списка в соответствии с каноническим порядком. Очевидно, что мы имеем:

**Теорема 2** CanonicalBest *имеет коэффициент аппроксимации 2*.

*Доказательство*: алгоритм PseudoBatch имеет коэффициент аппроксимации 2 и является приоритетным алгоритмом. Учитывая пример проблемы, алгоритм CanonicalBest выдает график со средневзвешенным завершением не хуже, чем алгоритм PseudoBatch.

**3 Альтернативный линейный алгоритм для пакетного списка**

Теперь перейдем к времени выполнения алгоритма CanonicalBest. Как упоминалось ранее, Брукер и Альберс [1] дали алгоритм линейного времени для задачи пакетной обработки списка. Здесь мы приведем простой альтернативный алгоритм. Алгоритм подобен их алгоритму, но в качестве подпрограммы он использует хорошо известный алгоритм линейного поиска по матрице времени Лармора и Шибера [15].

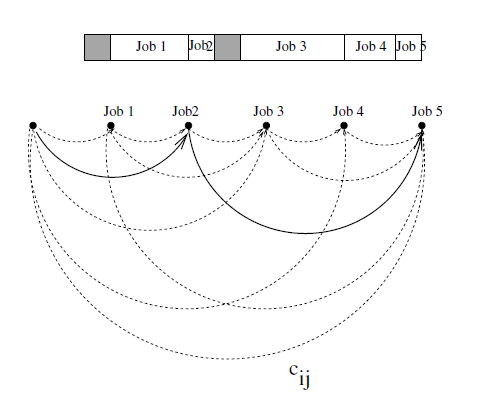
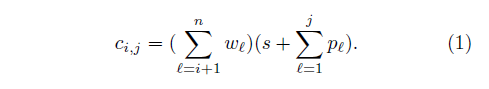


Рисунок 4 - Сведение проблемы пакетной обработки списка к проблеме пути.

Для этого предположим, что задания равны 1, ... , *n* и заданы в этом порядке. Затем можно свести проблему пакетной обработки списка к задаче кратчайшего пути следующим образом: построить взвешенный направленный ациклический граф *G* с узлами *i* = 1, ... , *n* (то есть один узел для каждого задания) и добавить фиктивный узел 0 . Существует ребро (*i*, *j*), если *i* <*j*. (см. схему на рис. 4). Пусть ребра для *i* <*j* определяются как



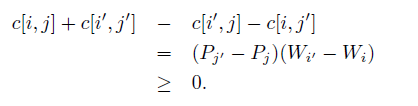
Кратко отметим:

**Лемма 3** *Матрица C = (), определенная в (1), является матрицей Монжа для всех вариантов чисел , ≥ 0. Более того, значения могут быть запрошены за O(1) время после линейной предварительной обработки.*

Доказательство. Пусть и - частичная сумма значений и . Тогда у нас есть



Для i <i' и j <j'



Кроме того, обратите внимание, что эти значения могут быть запрошены за O(1) время после линейной предварительной обработки, настроив массивы частичных сумм для и за линейное время.

Возвращаясь теперь к обсуждению редукции, легко увидеть (подробнее см. [1]), что стоимость пути <0, , , … , , *n*> дает значение расписания, которое пакетируется при каждом задании , , … , . И наоборот, любое пакетипование со стоимостью *A* соответствует пути в *G* с длиной пути *A*.