Алгоритмы для задач разбиения и упаковки с неполной информацией

В. М. Котов*

Полуонлайновые (semi-online) алгоритмы представляют собой промежуточный подход, призванный повысить практическую применимость моделей онлайн- и офлайн-планирования. Полуонлайновое планирование — это модифицированная версия онлайн-планирования, в которой, помимо входных данных, доступна дополнительная информация (Extra Piece of Information, EPI) или имеется буфер для хранения данных. В работе дается введение в онлайн- и полуонлайновые подходы для задач разбиения (планирования в многопроцессорных системах), и упаковки, обсуждаются области их применения и мотивация исследований, а также предлагается общая классификация моделей полуонлайнового планирования. Приводятся основные подходы при разработке и анализе алгоритмов.

Ключевые слова: онлайн версия, полуонлайн версия, сравнительный анализ, гарантированная оценка, нижняя граница.

1. Задачи разбиения и упаковки с неполной информацией

Задачи разбиения и упаковки относятся к классическим задачам дискретной оптимизации и являются модельными для широкого круга прикладных проблем. Известно, что даже простейшие задачи, связанные разбиением и упаковками, относятся к классу NP-трудных проблем, и неизвестно, существуют ли для них полиномиальные алгоритмы. Практическая важность таких задач обуславливает необходимость их решения за приемлемое время с гарантированной точностью, поэтому актуальна разработка и анализ эффективных приближенных алгоритмов.

В последнее время большой интерес вызывают версии задачи с неполной информацией, так называемые онлайн и полу-онлайн версии.

- В онлайн-модели входные данные поступают последовательно, один за другим. Каждый поступивший элемент должен быть обработан немедленно, без какой-либо информации о последующих данных.

* *Котов Владимир Михайлович* — д.ф.-м.н, профессор Белорусского государственного университета, e-mail: kotovvm@bsu.by, ORCID: 0000-0002-1022-875X.

Kotov Vladimir Mikhailovich — doctor of sciences, professor of Belorussian State University.

Опубликовано на условиях лицензии CC-BY 4.0
Published under CC-BY 4.0

В онлайн-модели на шаге t известна последовательность входных данных $I_t = \langle i_1, i_2, \ldots, i_{t-1}, i_t \rangle$; она должна быть обработана окончательно и бесповоротно, при этом информация о дальнейшей последовательности $\langle i_{t+1}, \ldots, i_{n-1}, i_n \rangle$ отсутствует (где $t \geq 1$). Алгоритм, предназначенный для работы в таких условиях, называется онлайн-алгоритмом. Онлайн-алгоритм формирует промежуточный результат o_t для каждого входного элемента i_t (где $1 \leq t \leq n-1$) непосредственно в процессе поступления данных, прежде чем выдать окончательный результат o_n.

- В полуонлайн-модели (semi-online framework) входные данные, как и в онлайн-модели, поступают поочередно, однако при этом доступна дополнительная информация (Extra Piece of Information, EPI) о будущих входных данных. На любом шаге t полуонлайн-алгоритм получает последовательность входных данных I_t вместе с дополнительной информацией (EPI) и обрабатывает их окончательно и бесповоротно, формируя промежуточный результат o_t (где $1 \leq t \leq n-1$), прежде чем выдать итоговый результат o_n.

Полуонлайновый подход (semi-online approach) представляет собой промежуточную модель, призванную учесть как практическую применимость, так и ограничения, свойственные чисто онлайновым и офлайновым схемам. В большинстве реальных ситуаций входные данные не доступны в полном объеме с самого начала, но при этом они не обязательно поступают исключительно в онлайновом режиме — возможен поэтапный приток данных, сопровождающийся получением дополнительной информации о последующих элементах. Например, в сервисе потокового видео запросы на скачивание файлов поступают динамически, однако системе заранее известны такие параметры, как наиболее востребованный видеофайл или файл наибольшего размера.

Близкой к полуонлайновой модели является так называемая «модель с подсказками» (advice model), где дополнительная априорная информация (EPI) представлена в виде «подсказок» (битов информации).

2. Практическое применение

Рассмотрим некоторые важные области, в которых полуонлайновое планирование служит ключевой алгоритмической основой.

- Управление ресурсами в операционных системах: в многопользовательских операционных системах с разделением времени заранее неизвестно, в каком порядке будут поступать задания и каково

будет их общее количество. В таких системах задания передаются планировщику по мере их поступления. Одной из неотъемлемых функций планировщика является прогнозирование максимального и минимального времени, необходимого для выполнения поступившей задачи. Цель состоит в том, чтобы сразу после появления задачи закрепить за ней необходимые вычислительные ресурсы (например, память или процессорное время) — причем сделать это окончательно и бесповоротно, — стремясь тем самым минимизировать время ее выполнения.

- Распределенное управление данными: В распределенных и параллельных системах часто возникает задача размещения файлов разного размера на удаленных серверах с ограниченной емкостью. Файлы поступают из известного источника динамически, и каждый из них должен быть немедленно распределен на один из удаленных серверов. При этом центральный планировщик системы, принимая окончательное решение о размещении, не обладает информацией о том, какие файлы могут поступить в дальнейшем. Однако известно, что источник хранит файлы на серверах емкостью k единиц; это дает некоторое представление об общем объеме данных, которые предстоит разместить. Задача заключается в том, чтобы распределить файлы по удаленным серверам, минимизируя при этом требуемый объем хранилища.

- Управление запросами к серверу или веб-кэширование: В модели «клиент-сервер» заранее неизвестно ни количество запросов, которые поступят на удаленные серверы, ни время, необходимое для их обработки. Иерархическая организация серверов может служить дополнительным источником информации для планировщика, позволяя предоставлять различные уровни обслуживания для разных запросов, исходя из общей цели — выполнить каждый запрос как можно позже (но в установленные сроки).

- Производство: заказы от клиентов поступают в производственную систему в режиме реального времени. Ресурсы — персонал, оборудование и производственные мощности — должны распределяться сразу после получения каждого заказа, при этом информация о будущих заказах отсутствует. Тем не менее, можно оценить минимальное или максимальное время, необходимое для выполнения заказа. Поступление заказов в режиме реального времени существенно влияет на решения об аренде или приобретении дорогостоящего оборудования для производственных подразделений.

- Техническое обслуживание и модернизация промышленного оборудования: планирование различных видов технического обслуживания и эксплуатационных работ для модульных газотурбинных авиадвигателей. Цель состоит в том, чтобы распределить задачи между единицами оборудования таким образом, чтобы сбалансировать нагрузку на них. На практике часто стремятся максимизировать загрузку того оборудования, которое на данный момент загружено меньше всего.

3. Показатели эффективности для полуонлайнового планирования

Традиционные методы анализа эффективности офлайновых алгоритмов планирования в значительной степени опираются на знание всей последовательности задач. В связи с этим они малопригодны для оценки полуонлайновых алгоритмов, которые обрабатывают поступающие данные пошагово, обладая лишь минимальной информацией о будущих запросах. Метод конкурентного анализа позволяет оценить эффективность полуонлайнового алгоритма (ALG) в наихудшем случае — как для задач минимизации, так и для задач максимизации — путем вычисления конкурентного отношения (competitive ratio, CR). Для задачи минимизации стоимости конкурентное отношение определяется как наименьшее положительное число k ($k \geq 1$), такое что для любой допустимой последовательности входных данных из множества $I = \{i_1, i_2, \ldots, i_n\}$ выполняется неравенство $C_{ALG} \leq k \cdot C_{OPT}$, где C_{ALG} — стоимость, полученная алгоритмом ALG для данной последовательности, а C_{OPT} — оптимальная стоимость, достижимая с помощью идеального офлайнового алгоритма (OPT).

Верхняя граница (upper bound, UB) конкурентного отношения для алгоритма ALG задает максимально возможное значение CR для всех допустимых последовательностей из I. Нижняя граница (lower bound, LB) конкурентного отношения для полуонлайновой задачи X означает, что существует такой набор входных данных I, при котором любой полуонлайновый алгоритм ALG неизбежно получит стоимость $C_{ALG} \geq b \cdot C_{OPT}$, где b — нижняя граница для задачи X. Эффективность алгоритма ALG считается «точной» (tight), если между полученными нижней и верхней границами для рассматриваемой задачи нет разрыва. В ряде случаев говорят, что эффективность алгоритма ALG является точной, если выполняется равенство $C_{ALG} = k \cdot C_{OPT}$. Для задачи максимизации стоимости конкурентное отношение определяется как точная нижняя грань

(infimum) значений k, таких что для любой допустимой последовательности входных данных из I выполняется неравенство $k \cdot C_{ALG} \geq C_{OPT}$. Цель полуонлайнового алгоритма — добиться значения конкурентного отношения, максимально приближенного к 1 (при условии, что оно всегда больше или равно 1).

Дополнительная информация (EPI) — это сведения о будущих данных, предоставляемые алгоритму онлайн-планирования. В научной литературе рассматривается ряд таких видов дополнительной информации, использование которых позволяет существенно повысить эффективность по сравнению со стандартными алгоритмами онлайн-планирования. Ниже приведены определения и обозначения некоторых хорошо изученных типов EPI:

- Сумма (Sum, T): $\sum_{i=1}^{n} p_i$ — суммарный объем всех задач.
- Максимум (Max, $p_{\max}$): $\max\{p_i \mid 1 \leq i \leq n\}$ — наибольшее время обработки или наибольший объем задачи.
- Оптимальное время завершения (Opt): значение оптимального времени завершения (makespan) часто оценивается с помощью двух основных границ: $C_{OPT} \geq \frac{1}{m} \sum_{i=1}^{n} p_{ij}$ и $C_{OPT} \geq p_{\max}$.
- Ограниченные диапазоны времени обработки (TGRP): наличие нижней и/или верхней границ для времени обработки всех задач, либо нижнюю границу $TGRP(lb)$, либо верхнюю границу $TGRP(ub)$ для времени обработки задач.
- Порядок поступления задач: $p_{i+1} \leq p_i$ для $1 \leq i \leq n$ — задачи поступают в порядке невозрастания их объема (Decr); либо $r_{i+1} \geq r_i$ — задачи поступают в порядке неубывания моментов их появления (Incr-r).
- Буфер ($B(k)$): накопитель конечного размера k (≥ 1), способный хранить не более k задач. Общий объем задач в буфере $B(k)$ составляет $w(B(k)) \leq \sum_{i=1}^{k} p_i$. Наличие буфера позволяет алгоритму онлайн-планирования либо временно поместить поступившее данное в буфер, либо — если буфер заполнен — окончательно назначить задачу на конкретный исполнительный узел (машину). Таким образом, перед принятием эффективного решения о планировании алгоритму всегда известны параметры $k + 1$ будущих данных. В научной литературе рассматривались следующие варианты длины и использования буфера: буфер длиной k (≥ 1), обозначаемый как $B(k)$; буфер длиной 1, обозначаемый как $B(1)$; а также буфер с возможностью изменения порядка элементов, обозначаемый как $reB(k)$. - Информация о последней задаче. Заранее известно,

что последняя задача имеет наибольшее время обработки, то есть $p_n = p_{max}$; этот тип EPI обозначается как LL. Рассматривается случай, когда несколько задач поступают одновременно с последней задачей; такой тип дополнительной информации обозначается как Sugg.

- Неточная частичная информация. Неточная частичная информация (также называемая «искаженной» частичной информацией) относится к ситуации, когда дополнительные сведения, доступные онлайн-алгоритму, не являются абсолютно точными. Например, алгоритму известно лишь приближенное значение некоторой величины (например, суммы), а не ее точное значение. Этот тип дополнительной информации обозначается как disSum.
- Перераспределение задач (reasgn). После того как все задачи уже распределены по машинам, их можно перераспределить между ними, следуя определенным правилам. Предложено несколько стратегий перераспределения: перераспределение последних k задач (обозначается как reasgn(last(k))); перераспределение произвольных k задач (reasgn(k*)); перераспределение только последних задач на всех машинах (reasgn(last)*); или перераспределение последней задачи на одной из машин (обозначается как reasgn((last)1)).
- Доступность машин (mchavl). Машины могут быть доступны не с самого начала. Они становятся доступными по мере необходимости, причем время готовности (r_j) для каждой машины M_j известно заранее. Также рассматривается сценарий, при котором одна машина доступна для выполнения любых задач, а другие — только для определенных, заранее заданных задач.
- Уровень обслуживания (Grade of Service, GOS) или иерархия машин. Априори известно, что машины организованы в иерархическую структуру для обеспечения различных уровней обслуживания задач в соответствии с заданным GOS. Например, если для задачи J_i установлен уровень обслуживания g_2, то она может быть назначена только на машину M_2; если же для J_i установлен уровень g_1, то она может быть запланирована на любой из доступных машин.

4. Основные подходы для решения версий задач с неполной информацией

Вначале основным подходом для решения таких задач являлись алгоритмы градиентного типа. Позже появились исследования, использующие

комбинаторные и структурные особенности задач для уточнения нижних оценок оптимального решения, что является ключевым фактором для определения гарантированной оценки предлагаемого алгоритма.

Авторами была предложена техника формирования групп множеств (процессоров, контейнеров) и классификации весов предметов, методика пересчета нижних оценок оптимального решения, при которых обеспечивается конечное число инвариантов при произвольном порядке поступления данных. Основная идея такого подхода состоит в том, что множества и предметы разбиваются на группы, позволяющие поддерживать некоторые инварианты, обеспечивающие требуемую гарантированную оценку алгоритма.

Использование такого подхода позволило получить рекордные по точности алгоритмы для нескольких версий задач.

5. Пример

Дополнительная информация: Известная сумма элементов.

Критерий: минимизировать максимальную сумму в разбиении на m множеств (процессоров).

Будем считать, что Z — оптимальное значение целевого функционала, а S — общая сумма длительностей выполнения работ для полуонлайн-версий задачи. Тогда нижняя граница LB для оптимального значения функционала очевидным образом.

В первом случае она равна Z, а во втором — средней загрузке процессоров S/m.

Не умаляя общности, будем считать, что $\mathrm{LB} = 1$. Этого можно добиться, пронормировав все величины (поделив на значение LB). Определим параметр $\alpha > 0$ и, используя этот параметр, опишем алгоритм, гарантированная оценка которого будет $(1 + \alpha)$. Разобьем поступающие работы на классы в соответствии с их процессорным временем.

В класс A определим работы, у которых процессорное время больше чем $(1 + \alpha)/2$. Для того чтобы обеспечить нужную гарантированную оценку, очевидно, что на один процессор не надо назначать две работы из класса A.

В класс B определим работы, у которых процессорное время больше α, но не больше чем $(1 + \alpha)/2$. Для того чтобы обеспечить нужную гарантированную оценку, очевидно, что на один процессор можно назначать две работы из класса B. При этом загрузка такого процессора будет больше 1.

В класс C определим работы, у которых процессорное время не больше чем α. Добавление таких работ на процессор с загрузкой меньше LB

приведет к тому, что загрузка процессора превысит 1, но не превысит $(1 + \alpha)$.

Понятно, что в любом решении невозможна ситуация, когда загрузка каждого процессора превосходит 1. Основная идея алгоритма состоит в такой загрузке процессоров, чтобы для одного процессора или группы процессоров их средняя загрузка была не меньше 1, а максимальная загрузка любого из процессоров не превосходила более чем в $(1 + \alpha)$ раз величину оптимального значения целевой функции.

Ключевым моментом данной схемы является выделение специальных типов процессоров. К первому типу относятся процессоры, на каждый из которых назначена одна работа из класса A и несколько работ из класса C, причем их общая сумма превышает 1, но не превышает более чем в $(1 + \alpha)$ раз оптимальное значение целевой функции.

Ко второму типу относятся процессоры, на каждый из которых назначена одна работа из класса A и, возможно, несколько работ из класса C, причем их общая сумма меньше 1.

К третьему типу относятся процессоры, на каждый из которых назначены две работы из класса B. Возможно наличие одного процессора с одной работой из класса B, если количество таких работ нечетно.

Процессоры четвертого типа разбиты на группы из четырех процессоров каждая и обладают следующим свойством.

Свойство 1. Загрузка каждого из трех первых процессоров группы не превосходит α, четвертый процессор пуст, а суммарная загрузка группы — как минимум $3\alpha/2$.

Алгоритм построения расписания состоит из двух этапов. На первом этапе формируются группы одного из типов следующим образом.

1. Если поступает работа из класса A, то приоритетным для нее является наиболее загруженный процессор из группы типа 4. Иначе работа назначается на пустой процессор.
2. Если поступает работа из класса B, то приоритетным для нее является наиболее загруженный процессор из группы типа 3. Иначе работа назначается на пустой процессор.
3. Если поступает работа из класса C, то приоритетным для нее является наиболее загруженный процессор типа 2. Если таких процессоров нет, то работа назначается на процессор из завершенной группы типа 4. При невозможности формируется новая группа типа 4.

Очевидно, что после первого этапа не могут быть одновременно получены группы процессоров типа 2 и типа 4, а для каждой группы процессоров четвертого типа выполняется свойство 1.

Идея второго этапа алгоритма основывается на использовании резервного четвертого процессора в каждой группе четвертого типа, что позволяет избежать коллизий при поступлении работ из классов A и B.

Свойство 2. При назначении любой работы типа 1 (даже со значением длительности больше 1 в случае задачи 2) на процессор с загрузкой не более α новая суммарная загрузка этого процессора не превосходит более чем в $(1+\alpha)$ раз оптимального значения целевой функции.

Теорема 1. *При наличии непустой группы процессоров типа 2 при $\alpha = 2/3$ для любой последовательности длительностей работ при назначении работы с длительностью больше α на наименее загруженный процессор алгоритм работает корректно.*

Замечание. Для полуонлайн-версии, когда известно оптимальное значение целевой функции, при отсутствии процессоров типа 4 после первого этапа будут поступать только работы из класса C.

Теорема 2. *При наличии непустой группы процессоров типа 4 при $\alpha = 0{,}58505$ для любой последовательности длительностей работ существует алгоритм, который работает корректно.*

Следствие 1. *Для задачи, когда известно оптимальное значение целевой функции, построен алгоритм с гарантированной оценкой* 1,58505.

Отметим, что для задачи с известной общей суммой доказана нижняя граница, которая лежит в пределах интервала [1,58504; 1,58505].

Статья поступила 11 ноября 2026 г.

Algorithms for Partitioning and Packing Problems with Incomplete Information

V. M. Kotov

Semi-online algorithms represent an intermediate approach designed to enhance the practical applicability of online and offline scheduling models. Semi-online scheduling is a modified version of online scheduling in which, in addition to the standard input data, extra information (an "Extra Piece of Information," or EPI) is available, or a buffer is provided for data storage. This paper introduces online and semi-online approaches to partitioning (scheduling in multiprocessor systems) and bin-packing problems, discusses their areas of application and the motivations behind such research, and proposes a general classification of semi-online scheduling models. Furthermore, it outlines the primary methods used in the development and analysis of these algorithms.

Keywords: online version, semi-online version, comparative analysis, guaranteed estimate, lower bound.

Received on November 11, 2026