УДК 656.6

Трухина Мария Александровна [1], ст. преподаватель, e-mail: mat-vsuwt@ya.ru

[1]Волжский государственный университет водного транспорта, г. Нижний Новгород, Россия.

ПОДХОДЫ К РЕШЕНИЮ ЗАДАЧ ОПТИМИЗАЦИИ ОБСЛУЖИВАНИЯ ПОТОКОВ ПАКЕТОВ ОБЪЕКТОВ ТРАНСПОРТНОГО ТИПА МЕТОДОМ ДИНАМИЧЕСКОГО ПРОГРОАММИРОВАНИЯ

Аннотация. В статье рассматривается особенности реализации алгоритма динамического программирования для задач обслуживания конечного детерминированного потока пакетов объектов. Перечисляются возникающие проблемы и приводятся подходы к их решению.

Ключевые слова: динамическое программирование; вычислительная сложность; задача транспортного типа; одностадийное обслуживание; поток объектов; переналадки; директивные сроки.

Введение

Рассматривается задача обслуживания конечного детерминированного потока пакетов объектов [1] в наиболее общей постановке. Задача потока обслуживания пакетов объектов является дальнейшим развитием задачи обслуживания конечного детерминированного потока объектов в системах транспортного типа и является NP-трудной [2]. На практике это означает отсутствие алгоритмов поиска оптимального решения, существенно лучших, чем полный перебор. Стандартными способами сокращения перебора являются подходы, основанные на идеологии ветвей и границ, и динамического программирования [3]. Методы, основанные на подходе ветвей и границ, часто обладают непредсказуемой длительностью отработки решающего алгоритма. В отличие от них, методы основанные на динамическом программировании (ДП) демонстрируют более предсказуемое поведение.

Данная статья описывает проблемы, возникающие при реализации алгоритма динамического программирования алгоритма ДП для указанной задачи с модификациями переналадки (мультипоток) [4] и директивные сроки [5] , и подходы к их решению.

Постановка задачи

Процессор P должен выполнить обслуживание мультипотока M объектов, прибывающих в составе пакетов $O(s)$, $s=\overline{1,h}$. Каждый пакет $O(s)$ состоит из объектов $o_{s,1}$, $o_{s,2}$,…, $o_{s,n(s)}$, подлежащих однократному без прерываний обслуживанию. Общее количество объектов в мультипотоке N.

Для каждого пакета $O(s)$ считается известным момент t_s его поступления в очередь для обслуживания процессором P. Обслуживание объектов процессором P может

выполняться в произвольном порядке, т.е без учета принадлежности их тому или иному пакету. Длительность обслуживания объекта $o_{s,l}$ определяется значение параметра $\tau_{s,l}$.

Каждый объект $o_{s,l}$ характеризуется величиной g_i – признак принадлежности объекта к подпотоку M_{g_i}, $g_i = \overline{1,m}$. При переходе на обслуживание объекта другого подпотока требуется переналадка. Длительности переналадки задаются квадратной матрицей *H*. Обслуживание каждого объекта реализуется процессором без прерываний; в каждый момент времени он может обслуживать только один объект; по завершению обслуживания объект немедленно освобождает процессор; немотивированные простои процессора и объектов запрещены.

Пакет объектов $O(s)$ считается обслуженным, если завершены обслуживанием все входящие в его состав объекты.

Момент t_s^f завершения обслуживания пакета $O(s)$ определяется как $\max(t_{s,1}^f, t_{s,2}^f, ..., t_{s,n(s)}^f)$, где $t_{s,l}^f$ – момент завершения обслуживания объекта $o_{s,l}$. За время от момента прибытия до момента завершения обслуживания по пакету $O(s)$ налагается штраф $Fs(\Delta)$. Монотонная неубывающая функция $Fs(\Delta)$ именуется функцией индивидуального штрафа по пакету $O(s)$.

Стратегия обслуживания записывается как произвольная перестановка S индексов всех объектов, т.е. элементов мультипотока M. При известной стратегии S для каждого объекта моменты начала и завершения обслуживания вычисляются очевидным образом арифметически.

Рассматриваемые в докладе задачи записываются в виде.

1. Построить расписание обслуживания объектов мультипотока M, минимизирующее суммарный штраф по всем пакетам, т.е.

$$\min_S \sum_{s=1}^{h} F_s(t_s^f(S) - t_s). \qquad (1)$$

2. Построить расписание обслуживания объектов мультипотока M, минимизирующее значение максимального из индивидуальных штрафов по всем пакетам, т.е.

$$\min_S \left(\max_{s=1,2,...h} F_s(t_s^f(S) - t_s) \right). \qquad (2)$$

Рассмотренная модель обслуживания адекватно описывает процессы подачу к терминалу грузовой обработки многосекционных судовых составов.

Соотношения ДП

Приведем рекуррентные соотношения ДП для самого полного варианта

$$B(t, g, Z) = \min_o \begin{cases} \varphi_s + B(t_o^{кон}, g, Z \setminus \{o_{s,i}\} \cup D(t, t_o^{кон})), & \text{Если } o = o_{s,i},\ Z \setminus \{o_{s,i}\} \cap O(s) = \varnothing, \\ B(t_o^{кон}, g, Z \setminus \{o_{s,i}\} \cup D(t, t_o^{кон})), & \text{Если } o = o_{s,i},\ Z \setminus \{o_{s,i}\} \cap O(s) \neq \varnothing, \\ B(t_o^{кон}, j, Z \cup D(t, t_o^{кон})), & \text{Если } o = f_{g,j}, \\ B(t_o^{кон}, g, Z \cup D(t, t_o^{кон})), & \text{Если } o = f. \end{cases} \qquad (3)$$

Здесь $B(t, g, Z)$ – функция Беллмана, параметры - время принятия решения, текущий подпоток, множество объектов, ожидающих обслуживания; φ_s – штраф за обслуживание

пакета s, $t_o^{кон}$ – время завершения обслуживания объекта o, D(t, t+Δ) – объекты, поступающие в систему на интервале времени [t+1, t+Δ], $f_{g,j}$ – переналадка (фиктивный объект), f – ожидание следующего (фиктивный объект).

Сложность задачи

Базовая рассматриваемая задача – линейный критерий, поток объектов. Задача является NP трудной.

Сложность реализации исследуемых алгоритмов определяется рассмотрением нескольких независимых модификаций задачи.

Варианты:

1. Вариант критерия: линейный / минимаксный
2. Учет времени прихода объекта: да (поток объектов) / нет (множество)
3. Объединение объектов в пакеты: да (объекты объединены в пакеты, штраф начисляется за время обслуживания пакета) / нет (поток объектов, штраф по каждому объекту)
4. Наличие переналадок: да (объекты распределены по подпотокам, между подпотоками требуется переналадка) / нет
5. Наличие директивных сроков: да/ нет

Каждый вариант может быть применен независимо, что дает 2^5 итого 32 варианта задач.

Наличие 32 вариантов программы весьма трудоёмко с точки зрения отладки, тестирования и дальнейшего использования.

В связи с этим ставится цель на выходе получить одну программу.

Основная проблема – поддержка многовариантности. Для этого было выполнено следующее:

1. Стандартизован формат входного файла
2. Программа управляется ключами командной строки
3. Тип задачи кодируется строкой в параметрах
    a. (SDPTM) подпотоки / директивные сроки / пакеты / время прихода / минимаксный критерий,
    b. В любых комбинациях

Сложность программы

Сложность задачи приводит к сложности программы. Для борьбы со сложностью применены

1. Декомпозиция
   Верхний уровень: решение состоит из трех проектов
    a. Классы, представляющие предметную область
        - Поток пакетов
        - Пакет объектов
        - Ведущий объект
        - Ведомый объект
    b. Блок тестирования этих классов
    c. Блок решателя

   Используемый результат —
   EXE решателя + DLL классов
2. Автоматизированное тестирование

Позволяет быстро проводить регрессионное тестирование
(время прогона тестов после изменения — две минуты)

- Стандартизован формат входного файла
- Имеется генератор тестов
- Имеются тестовые наборы
- Имеются файлы для запуска пакетов тестов

3. Система контроля версий
   Позволяет
   - Производить тестирование без опасения «испортить» проект
   - Иметь возможность отследить изменения
   - В любой момент выделить одну из предыдущих версий
   - Иметь несколько активных веток
   - Выделить изменения и применить на другой ветке

Большое потребление памяти

Исходная задача является NP трудной. Применение динамического программирования позволяет сократить факториальное пространство поиска до $O(N*2^N))$

При этом решение динамическим программированием для нахождения оптимального значения целевой функции требует заполнения / хранения таблиц значений функции Беллмана (размер порядка 2^N x Tmax, можно оценить как $O(N*2^N)$).

Модификации с M подпотоками увеличивают требования к памяти в M раз.

Кроме того, необходимость реконструкции оптимального значения на обратном проходе динамического программирования требует для каждого значения таблицы хранить выбор оптимального шага (еще столько же чисел).

Для уменьшения требований программы к памяти были опробованы следующие подходы:

1. Хранение таблиц выбора оптимального шага.

Не хранить в памяти таблицы выбора оптимального шага. Вместо этого, в процессе восстановления оптимальной стратегии обслуживания осуществлять процесс выбора, идентичный процессу выбора в ходе прямого прохода динамического программирования.

Для этого блок процесса выбора был выделен в функцию, используемую при прямом и обратном проходах ДП.

Данная модификация позволила сэкономить половину памяти, таким образом, для задач без подпотоков, практически повысив размер решаемых задач на единицу.

2. Оптимизация структуры хранения значений функции Беллмана.

2.1. «Ступенчатый» массив

Функция Беллмана.имеет вид $B(t, g, Z)$ (параметры - время принятия решения, текущий подпоток, множество объектов, ожидающих обслуживания), соответственно место под хранение значений определяется количеством значений аргументов $B(t, g, Z)$. Самый большой вклад вносит составляющая Z – до 2^N. Нетрудно видеть, что если текущее время меньше времени прихода объекта, он никак не может быть ожидающим. Считая объекты упорядоченными по времени обслуживания, получаем число возможных аргументов Z для $t < t_N$ равным 2^K, K<N.

Это соображение можно использовать, если распределять массив построчно с разным числом элементов, зависящим от t.

Данная модификация была внедрена и, судя по контролю используемой алгоритмом памяти, дала положительный эффект.

2.2 Заполнение в процессе рекурсивного вызова

Следующая оптимизация касается алгоритма заполнения таблицы. Базовый метод ДП предполагает заполнение всей таблицы от «последних» значений параметра *t* вниз. Вариантом является заполнение таблицы посредством мемоизации при рекурсивном вызове функции Беллмана.

Данная модификация была внедрена. Просмотр заполненных таблиц после завершения алгоритма показал, что значительная часть элементов осталась неиспользованной (незаполненной) – самые заполненные строки использованы на 2/3. Даже если эта память не была сэкономлена, значения не были рассчитаны, таким образом ускоряя работу алгоритма.

2.3 Использование словаря

Предыдущая модификация показала нам, что в массиве имеются неиспользованные значения. Была предпринята попытка использовать более сложную структуры, а именно словарь.

Непосредственная попытка использовать словарь вместо массива не привела к успеху – накладные расходы на хранение данных существенно больше (в 4 – 10 раз), соответственно реальная располагаемая память упала, размер решаемой задачи упал.

Была предпринята вторая попытка.

Изначально на строку, соответствующую t, заводится словарь. По достижении словарем размера 10% от расчетной длины строки (2^K) словарь преобразовывается в обычный массив.

Этот подход показал свою работоспособность, размер решаемой задачи достиг размера задачи решаемой с массивом. Но выигрыша не получилось – шага вперед по размерности не случилось, более сложный алгоритм работал на 50% дольше.

Данная модификация была отвергнута.

Результаты

Результатом данной работы является реализация алгоритма ДП для рассмотренной задачи и всех рассмотренных ее модификаций в виде одного файла-решателя с интерфейсом командной строки и выводом в стандартный вывод, предназначенная для дальнейшего тестирования и/или использования в расчетной системе оптимизации обслуживания.

Список литературы:

1. Модели и оптимизационные задачи однопроцессорного обслуживания пакетов объектов/ Коган Д.И., Трухина М.А., Федосенко Ю.С. и др.// Автоматика и телемеханика. РАН. 2016. № 11. С. 142-157.
2. Коган Д.И., Федосенко Ю.С. Задача диспетчеризации: анализ вычислительной сложности и полиномиально разрешимые подклассы // Дискретная математика. – 1996. Т. 8. Вып. 3. – С. 135–147.
3. Беллман Р., Дрейфус С. Прикладные задачи динамического программирования. М.: Наука, 1965. 460 с.
4. Модель и алгоритм синтеза стратегий обслуживания мультипотока объектов мобильным процессором / Коган Д.И., Трухина М.А., Федосенко Ю.С. и др. // Системы управления и информационные технологии. – 2015. – №2(60). – С.40–43
5. Подходы к учету переналадок в задаче однопроцессорного обслуживания мультипотока объектов/ Трухина М.А., Шеянов А.В. //Дискретные модели в теории управляющих систем: X Международная конференция, Москва и Подмосковье, 23–25 мая

2018 г. : Труды / Отв. ред. В. Б. Алексеев, Д. С. Романов, Б. Р. Данилов. – Москва: МАКС Пресс, 2018.

SOLVING PROBLEMS OF OPTIMAL SERVICING OF FLOWS OF TRANSPORT-TYPE OBJECT PACKETS BY DINAMIC PROGRAMMING

Maria A. Trukhina

Abstract. The article discusses the features of the implementation of the dynamic programming algorithm for the tasks of servicing a finite deterministic flow of object packets. The problems that arise are listed and approaches to their solution are given.

Keywords: dynamic programming, computational complexity, transport-type problem, single-stage servicing, object flow, reconfiguring, deadlines.