УДК 519,12.176

РАЗРАБОТКА ЭФФЕКТИВНОГО АЛГОРИТМА ВЫЧИСЛЕНИЯ ДЛИНЫ ДОПУСТИМОГО РАСПИСАНИЯ УПОРЯДОЧЕНИЯ ТРЕХЭТАПНЫХ РАБОТ В СИСТЕМЕ ПОТОЧНОГО ТИПА

Л.Н. Козачок, ассистент, ХНАДУ

Аннотация. *Исследуются вопросы применения математических средств теории расписаний для многостадийных вычислительных систем. Предлагается детерминированная математическая модель с набором независимых заданий, для которых выполняется задача упорядочения по критерию минимизации длины расписания.*

Ключевые слова: *теория расписаний, вычислительная система, расписание без прерываний.*

РОЗРОБКА ЕФЕКТИВНОГО АЛГОРИТМУ РОЗРАХУНКУ ДОВЖИНИ ПРИПУСТИМОГО РОЗКЛАДУ УПОРЯДКУВАННЯ ТРЬОХЕТАПНИХ РОБІТ У СИСТЕМІ ПОТОЧНОГО ТИПУ

Л.Н. Козачок, асистент, ХНАДУ

Анотація. *Досліджуються питання використання математичних засобів теорії розкладів для багатоетапних обчислювальних систем. Пропонується детермінована математична модель з набором незалежних завдань, для яких виконується задача упорядкування за критерієм мінімізації довжини розкладу.*

Ключові слова: *теорія розкладів, обчислювальна система, розклад без переривань.*

DEVELOPMENT OF EFFECTIVE LOGICAL DESING OF LENGTH OF ADMISSIBLE SCHEDULE OF THREE-STAGE WORKS ORDERING IN THE FLOW TYPE SYSTEM

L. Kozachok, assistant, KhNAHU

Abstract. *The issues concerning the application of mathematical means of scheduling theory for many-stage flow systems is investigated in the given article. The deterministic mathematical model with a set of independent tasks, for which the task of ordering according to the criterion of scheduling length minimization is carried out, is offered.*

Key words: *scheduling theory, computation system, non-preemptive schedule.*

Введение

Рассмотрим множество работ G, на котором задано отношение частичного порядка. Элементы этого множества (работы) обслуживаются набором процессоров, реализующих отдельные этапы их выполнения работ. В статье описывается алгоритм упорядочения по критерию минимизации длины расписания и минимизации времени прохождения работ в системе процессоров, состоящей из трех этапов.

Эта задача относится к классу задач построения расписаний минимальной длины для многопроцессорных систем и является частным их случаем, для которой составляется алгоритм перебора для нахождения длины расписания прохождения работ в системе. Такого вида задачи могут использоваться в организации производственных процессов, в науке об

управлении, экономике, информатике, электротехнике и т.д.

Анализ публикаций

Задача минимизации общего времени обслуживания, известная в литературе как *(nM)*–задача Беллмана-Джонсона [1]. Подход, позволяющий исключить из рассмотрения заведомо неоптимальные расписания из числа тех, при которых не все процессоры обслуживают работы в одной и той же последовательности, предложен В.Ф. Золотухиным [2]. Задача максимизации функционала на множестве всех перестановок работ исследовалась Н.А. Лепешинским [3] и В. Шварцем [4].

Как показал В.С. Танаев [5], к задачам минимизации функционала на множестве всех перестановок вида сводятся задачи о нахождении оптимального расписания для многостадийных систем. Если не все процессоры обслуживают работы в одной и той же последовательности, то эти задачи могут оказаться *NP*-трудными [6]. Для задач построения оптимальных расписаний искомая перестановка может быть построена за операций.

Цель и постановка задачи

Целью данной работы является определение допустимого расписания функционирования системы обслуживания, имеющего наименьшую длину, что обеспечивает оптимальность по быстродействию на основе анализа алгоритма построения расписания.

Постановка задачи – вычислить длину расписания прохождения работ в системе обслуживания, состоящей из трех производственных этапов и включающей 1+*m*+1 процессоров.

Методика

Исследуется ситуация, когда определенное конечное множество работ (под работой понимается определенное количество заданий, деталей, требований) обслуживается в системе, реализующей три этапа выполнения работы. На каждом этапе работа обслуживается процессором или совокупностью процессоров; при этом, выполнение конкретной работы некоторым процессором называется операцией.

В нашем случае имеется конечное множество из n независимых работ , которые одномоментно поступают в систему для обслуживания на трех последовательных этапах: первый и третий этапы представлены одним процессором: соответственно, а второй этап – m параллельно действующими процессорами ,,$m<n$.

Длительности выполнения j работы заданы величинами на каждом этапе соответственно. Таким образом, работы, поступающие в систему, могут быть описаны тройками чисел .

В любой момент времени каждый процессор занят выполнением не более одной работы, а каждая работа выполняется не более чем на одном процессоре.Каждый последующий этап выполнения работы не может начаться раньше завершения предыдущего. Такие обслуживающие системы называются системами поточного типа.

Порядок прохождения процессоров, соответствующих определенному этапу, одинаков для всех работ, т.е. рассмотрим системы с одинаковыми маршрутами выполнения работ. Все отдельные этапы выполнения работы реализуются без прерываний.

Пусть второй этап выполнения работы осуществляется любым из процессоров . Для определенности будем полагать, что если к моменту завершения первого этапа некоторой работы несколько процессоров второго типа окажутся не загруженными, то второй этап будет выполняться на том процессоре, где раньше выполнилась одна из предыдущих работ, а именно на том, который дольше простаивает. Очевидно, *i*-й

процессор не может быть свободен, если имеются работы, прошедшие первый этап и готовые к выполнению второго этапа.

В случае, когда к моменту завершения некоторой работы процессором , выполнены первые этапы нескольких работ, то эти работы назначаются для выполнения на процессор в порядке прохождения их через единственный процессор первого уровня.
Пусть π – порядок запуска работ на процессор первого уровня. Из приведенных выше требований следует, что порядок π определяет последовательность П назначения работ на процессоры второго уровня и очередность выполнения работ на каждом параллельно действующем процессоре второго уровня. Вообще говоря, последовательность выполнения первых этапов работ отличается от последовательности их выполнения на третьем этапе. Допустимое решение задачи может быть задано парой перестановок (π, П), где π – порядок следования первых этапов работ, а П – последовательность выполнения их третьих этапов. Поиск соответствующего алгоритма *A* построения расписания прохождения работ в системе приводит к разбиению множества на подмножеств , где – множества работ, выполняемых процессором . Для этих подмножеств выполняются следующие ограничения: , где .

Таким образом, можно сказать, что при применении к множеству независимых работ алгоритма , определяющего последовательность выполнения этапов работ, получаем расписание функционирования системы . Целевую функцию задачи можно представить как , где – моменты завершения выполнения работы процессором на третьем этапе. Задача применения эффективного алгоритма состоит в минимизации целевой функции, т.е. в ускорении наступления момента завершения обработки последней работы из рассматриваемой перестановки П на третьем этапе.

Найдем длину расписания какотрезок времени от момента поступления первой работы начальной перестановки π на процессор (начало отсчета времени в системе) до момента завершения обработки последней работы перестановки П процессором . Предположим, что порядок поступления работ на первом этапе задан упорядоченной последовательностью , члены которой – индексы работ, расположенные в порядке поступления на процессор . Определим моменты окончания выполнения работ на втором этапе в виде . Упорядочим элементы последовательности по неубыванию их значений, т.е. . Таким образом, порядок следования третьих этапов работ определяется перестановкой , в которой

Допустим порядок поступления работ на процессор первого уровня задан, т.е. на выполнение подаются работы в порядке следующей записи . После прохождения первого уровня, поочередно на процессоры второго уровня поступают первые *m* работ , т.к. все процессоры еще свободны.

Теперь рассмотрим нахождение длины допустимого расписания *W.* Для этого построим алгоритм *A*, который включает в себя следующие позиции:

1) Создание массивов.
   Создадим массив , элементы которого − моменты окончания вторых этапов работ, а первые m элементов , и массив индексов работ.
2) Определение процессора, который будет выполнять *(m+1)*-ю работу.
   Расположить по неубыванию первые m элементов массива , для этого выполнить:
   a) где − номер минимального элемента массива;
   b) ;
   c) если то , а
   d) если , то перейти к пункту 2b.

Таким образом, будет найден минимальный элемент массива , который нужно поставить на первое место в массиве *V*, а элемент, занимавший первое место, на место с но-

мером ; следует также поменять местами значения элементов *L(I)* и *L(NMIN)* массива *L*.

e) ;
,
;

f) , если , то перейти к пункту 2а.

Расположив в пункте 2 *m* элементов массива *V* по неубыванию, определим процессор, который будет выполнять (*m*+1)-ю работу. Это процессор с номером *L*(*1*), т.к. элемент *V*(*1*) минимальный в массиве *V*(*m*), т.е. момент завершения второго этапа выполнения работы самый ранний. Значит второй этап (*m*+1)-ой работы будет выполнять процессор .

3) Определение момента окончания второго этапа (m+1)-ой работы.
   Если , то – момент окончания второго этапа (*m*+1)-й работы тем же процессором, на котором выполнялась работа , т.к. индексу *L*(*1*) соответствует минимальный элемент *V*(*1*).
   Если , то
4) Итерационный шаг.
   Рассмотрим следующие работы, т.е. *m*=*m*+1.
   Если *m*<*n*, то перейти к пункту 2, иначе к пункту 5.

Таким образом, добавляя постепенно в массив *V* по одному элементу, мы получим массив моментов окончания вторых этапов работ размерности *n* .

5) Начальная длина расписания.
   Определим длину *U* расписания функционирования системы , положив изначально .
6) Вычисление длины расписания прохождения всех работ в системе.
   Окончательная длина расписания определяется в цикле по *k=1* до *n-1*, в котором мы производим сравнения: если , то , иначе . В результате последнего сравнения с *V(n)* получаем окончательную длину расписания.

Данный алгоритм позволяет найти длину любого расписания согласно ограничениям о порядке загрузки процессоров второго уровня.

Пример применения алгоритма.

Рассмотрим *n*=10, *m*=5

Найдем длину расписания, не меняя порядок поступления работ на процессор первого уровня.

π=(1, 2, 3, …,10).

1) Создание массивов.
   Создадим массив *V(n)*, первые 5 элементов которого , , и масив *L*(*n*), элементы которого – номера работ.
2) Определение процессора, который будет выполнять (*m*+1)-ю работу.
   Расположим *V*(*k*), по неубыванию, переобозначая индексы у элементов массива *V*.
   Получим:
   L(*1*)=2
   L(2)=1
   L(3)=3
   L(4)=4
   L(*5*)=*5*
   Значит второй этап работы будет выполняться на процессоре , так как *L(1)=2*.
3) Определение момента окончания второго этапа (m+1)-ой работы.
   , тогда ,
4) Итерационные шаги.
   m=*m*+1=6, перейдем к пункту 2, в котором расположим значения элементов *V*(1),*V*(2),…,*V*(6) по неубыванию, изменяя, соответственно построенной последовательности, номера элементов массива, а начальные индексы сохраняем, присваивая их значениям элементов массива *L(k),* , получим

   .

   Таким образом, вычисляя *V*(7), *V*(8), *V*(9), *V*(10) и повторяя пункты 2 и 3, получим

V(1)= *L(1)=2*
V(2)= *L(2)=1*
V(3)= *L(3)=3*
V(4)= *L(4)=4*
V(5)= *L(5)=8*
V(6)= *L(6)=6*
V(7)= *L(7)=9*
V(8)= *L(8)=7*
V(9)= *L(9)=5*
V(10)= *L(10)=10*

Мы получили моменты окончания прохождения работами процессоров второго этапа. Процессоры второго этапа раньше всего закончили выполнение работы , затем работ . Поэтому порядок следования третьих этапов работ определяется перестановкой *П=(2, 1, 3, 4, 8, 6, 9, 7, 5, 10).*

5) Начальная длина расписания.
   .
6) Вычисление длины расписания прохождения всех работ в системе.
   В цикле по *k=1* до *9* производим сравнения:
   k=1: т.к. U<V(2)
   (9<10), то
   ;
   k=2: U<V(3)
   (12<3)
   ;
   и так далее
   k=8: U>V(9)
   (23>21)
   ;
   k=9: U>V(10)
   (23.3>22)
   .

Выводы.

В данной работе составлен алгоритм нахождения длины расписания, т.е. максимального времени завершения обслуживания всех работ в системе от момента поступления первой при определенном расписании работы на процессор первого уровня. Эти результаты могут применяться в многоплановом вопросе оценки качества построенного расписания.

Модели теории расписаний непрерывно развиваются, совершенствуются, обретают определенную общность, полнее отражают конкретные ситуации календарного планирования.

Литература.

1) Конвей Р.В., Максвелл В.Л., Миллер Л.В. Теория расписаний.−М.: Наука. −360 с.
2) Золотухин В.Ф. Об исключении заведомо неоптимальных решений задачи Беллмана-Джонсона. // Техн. Кибернетика.−1979.− №1.− с. 44-51.
3) Лепешинский Н.А. Одна задача теории расписаний и максимум суммы линейных форм на множестве подстановок. // Серия физ.-мат. Наук. −1987.− №3.−с.51-54.
4) Szwarc W. Extreme solution of the two-machine flow-shop problem. // Nav. Res. Log. Quart. – 1991.−V. 28, № 1.− P. 103-114.
5) Задачи оптимального планирования и проектирования: сб. научн. тр.− Минск. ИТК, 1991− 151 с.
6) Левин В.И. Структурно-логические методы в теории расписаний. Пенза: Изд-во Пенз. гос. технол. акад.,− 2006.− 370 с.

Рецензент: заведующий кафедрой автомобильной электроники ХНАДУ, д.т.н., профессор Бажинов А.В.
Статья сдана 15.06.2010 г.