ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2024, № 1, С. 122–130

КОМПЬЮТЕРНЫЕ МЕТОДЫ

УДК 519.86

ПЛАНИРОВАНИЕ ВЫЧИСЛЕНИЙ МНОГОПРОЦЕССОРНОЙ СИСТЕМЫ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

© 2024 г. М.Г. Фуругян
ФИЦ ИУ РАН, *Москва*, *Россия*
e-mail: *rtsccas@yandex.ru*
Поступила в редакцию 20.06.2023 г.
После доработки 19.08.2023 г.
Принята к публикации 02.10.2023 г.

Рассматривается задача планирования вычислений в многопроцессорной системе для случая, когда в некоторые моменты времени поступают запросы на выполнение комплексов работ с известными характеристиками. Допускаются прерывания и переключения с одного процессора на другой. В первой постановке состав всех комплексов и характеристики заданий известны заранее. Во второй постановке эта информация становится известной только в момент поступления каждого запроса. Требуется определить, существует ли допустимое расписание для совокупного комплекса работ и построить его в случае положительного ответа. Исследована постановка, в которой помимо процессоров имеется невозобновляемый ресурс. Разработан полиномиальный алгоритм решения задачи, основанный на построении сетевой потоковой модели и поиске максимального потока.

Ключевые слова: построение расписания, многопроцессорная система реального времени, допустимое расписание, сетевая модель, максимальный поток, не возобновляемый ресурс

DOI: 10.31857/S0002338824010091, **EDN:** WJFLNB

SCHEDULING CALCULATIONS FOR A MULTIPROCESSOR SYSTEM IN REAL TIME

© 2024 M. G. Furugyan
Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, *Moscow*, *Russia*
e-mail: *rtsccas@yandex.ru*

The problem of scheduling computations in a multiprocessor system is considered for the case when, at some time instants, requests for the execution of job packages with known characteristics are received. Interrupts and switching from one processor to another are allowed. In the first formulation, the composition of all complexes and the characteristics of tasks are known in advance. In the second setting, this information becomes known only at the time of each request. It is required to determine whether there is an admissible schedule for the total set of jobs and build it in case of a positive answer. A setting is studied in which, in addition to processors, there is a non-renewable resource. A polynomial algorithm for solving the problem is developed, based on the construction of a network flow model and the search for the maximum flow.

Keywords: scheduling computations, multiprocessor real-time system, admissible schedule, network model, maximum flow, non-renewable resource

Введение. Главной отличительной чертой вычислительных систем реального времени является то, что каждый прикладной модуль должен выполняться в строго заданном временном директивном интервале и завершиться не позднее установленного заранее срока. Такие системы находят широкое применение в различных областях деятельности человека. Например, при проектировании, испытаниях и эксплуатации сложных технических объектов (самолеты, ракеты, электростанции), при проведении опытно-конструкторских работ, в гражданском и военном строительстве, при оценке запасов полезных ископаемых в месторождениях, при обработке больших массивов информации, при проектировании и функционировании транспортных и конвейерных систем, во многих других областях. При этом одна из основных

задач заключается в распределении ресурсов вычислительной системы между программными модулями и построении оптимального расписания их выполнения. Алгоритмам решения таких задач посвящено большое количество публикаций. Здесь можно отметить такие фундаментальные работы, как [1—3], в которых авторы изучают различные постановки (составление расписаний с прерываниями и переключениями с одного процессора на другой и без прерываний, задачи на быстродействие и на соблюдение директивных сроков, построение однопроцессорных и многопроцессорных расписаний). В [3] исследуются NP-трудные задачи быстродействия и минимизации максимального временного смещения для одного и нескольких приборов. Предлагается новый подход к поиску приближенных решений. В [4, 5] рассматривается методика построения оптимальных расписаний в задачах с нефиксированными параметрами (длительности, потребляемые ресурсы). Методика основана на использовании метода "ветвей и границ" и построении многогранников устойчивости решений. В [6—8] разработана методика проверки выполнения ограничений реального времени, заключающихся в том, что каждая работа должна выполняться в заданном директивном интервале. Проведенные исследования выполнены для многоядерной вычислительной системы реального времени и базируются на построении имитационной модели с применением обобщенных конечных автоматов с остановкой таймера. С помощью этой модели строится временная диаграмма работы системы, позволяющая осуществить непосредственную проверку выполнения ограничений реального времени. В [8] предложен псевдополиномиальный алгоритм решения задачи построения оптимального по быстродействию расписания исполнения заданий с логическими условиями предшествования. В этой задаче для каждого задания дан список его непосредственных предшественников, а также число завершенных непосредственных предшественников, необходимое для начала его выполнения. Задача сведена к циклической игре. В [9, 10] некоторые задачи планирования работ сведены к минимаксным задачам.

Указанные выше публикации посвящены распределению возобновляемых ресурсов (процессоров, машин, исполнительных механизмов, приборов, рабочих), т.е. ресурсов, которые могут использоваться многократно. В ряде публикаций исследуются вопросы распределения невозобновляемых ресурсов (финансы, топливо, электроэнергия, различные материалы, оперативная память ЭВМ, закрепленная за определенными программными модулями). В отличие от возобновляемых ресурсов, невозобновляемые повторно применяться не могут. В этой связи отметим работы [11, 12], в которых предполагается, что длительности заданий линейно зависят от величины выделенного им ресурса. В [13] исследована задача со смешанными типами ресурсов — возобновляемыми и невозобновляемыми. Рассматривается задача составления допустимого расписания с прерываниями в многопроцессорной системе в случае, когда заданы директивные интервалы, процессоры могут иметь произвольные производительности, имеется несколько типов невозобновляемых ресурсов, а длительности выполнения работ линейно зависят от выделенного им количества этих ресурсов. Построены полиномиальные алгоритмы, основанные на сведе́нии исходной задачи к потоковой в сети специального вида.

Отметим несколько интересных статей по планированию в промышленном производстве. Так, в [14] авторы исследовали методику совместного планирования развития производственных мощностей и составления расписания с учетом рыночных возможностей, а также детализировали интегрированную модель планирования мощности производства с несколькими дискретными и непрерывными вариантами изменения краткосрочной и среднесрочной мощности и разработали эвристический алгоритм, основанный на сведении исходной задачи к нелинейной смешанной целочисленной задаче. В [15] представлены некоторые вопросы в области планирования и контроля производства, разработана иерархическая архитектура планирования и управления производством. В [16] приведена двухуровневая система хранения одного продукта, с помощью которой региональный центр пополняет заказы нескольких независимых местных распределительных центров в течение установленного периода времени. Разработанная модель определяет значения цены продукта, обязательного времени пополнения и обязательного времени доставки, которые максимизируют ожидаемую общесистемную прибыль за заданный период с учетом затрат на хранение продукта и фиксированных затрат на оборудование.

В [17] исследована задача, в которой планирование осуществляется в два этапа: сначала определяется последовательность действий, затем в график вставляется время простоя, чтобы минимизировать сумму ранних и поздних затрат. Последовательность работ определяется с помощью эвристического метода, а задача вставки времени простоя решается с помощью линейного программирования для задания времени начала и окончания действий. В [18] приведена задача планирования сроков выполнения заказов, а также указаны компромиссы, которые следует учитывать при установлении этих сроков. Предложена модель, показывающая, как запланированное время выполнения операции зависит от стохастической изменчивости требований

к ресурсам для этой операции, а также от использования ресурсов, связанных с этой операцией. В [19] описана задача планирования работы и маршрутизации двух роботов, которые доставляют продукты в определенные места. Решена задача минимизации времени выполнения всех операций и возврата роботов в исходное положение. Доказана NP-трудность задачи. Решение основано на использовании целочисленного линейного программирования и генетического алгоритма, а также динамического программирования для оценки качества решений.

В настоящей статье исследуется задача планирования вычислений в многопроцессорной системе при следующих предположениях. В заданные моменты времени поступают запросы на выполнение комплексов работ с известными длительностями и директивными интервалами. Допускаются прерывания и переключения с одного процессора на другой. Рассматриваются две постановки задачи. В каждой из них моменты поступления запросов известны заранее. Однако в первой постановке состав всех комплексов и характеристики работ также известны заранее, и поэтому в этом случае планировать выполнение всех заданий можно до момента поступления первого запроса. Во второй постановке состав комплексов заданий и их характеристики становятся известными только в момент поступления каждого запроса. Тогда планировать выполнение работ возможно только после поступления соответствующего запроса, т.е. в режиме реального времени. В обеих постановках требуется определить, существует ли допустимое расписание для всей совокупности комплексов работ и построить его в случае положительного ответа. Рассмотрена также задача для случая, когда помимо процессоров задания используют невозобновляемый ресурс. При этом длительность выполнения задания является убывающей функцией от количества выделенного ему невозобновляемого ресурса. В отличие от [13] не предполагается, что эта функция будет линейной. Решение задачи основано на построении сетевой потоковой модели и поиске максимального потока.

1. Постановка задачи. В моменты времени τ_k поступают запросы на выполнение K комплексов работ (заданий): $W_k = \left\{w_k^1, w_k^2, \ldots, w_k^{r_k}\right\}$, $k = \overline{1,K}$, $\tau_1 < \tau_2 < \ldots < \tau_K$. Для этого в каждом интервале $\left[\tau_k, \tau_{k+1}\right]$ имеется m_k процессоров (τ_{K+1} — момент времени, после которого эти процессоры использоваться не могут). Все процессоры идентичные. Каждое задание w_k^i имеет следующие характеристики: $[b_k^i, c_k^i]$ — директивный интервал (работа w_k^i может исполняться только в этом интервале), $b_k^i \geq \tau_k$, t_k^i — его длительность, $t_k^i \leq c_k^i - b_k^i$, $i = \overline{1,r_k}$. При выполнении заданий допускаются их прерывания и переключения с одного процессора на другой, которые по предположению не требуют временных затрат. Кроме того, не допускается параллельное исполнение одного задания несколькими процессорами и одновременное выполнение нескольких работ одним процессором.

Рассматриваются две постановки задачи. В каждой из них моменты τ_k известны заранее. Однако в первой постановке состав всех комплексов W_k и характеристики входящих в них работ также известны заранее. Поэтому в данном случае планировать работу всех заданий можно до момента времени τ_1. Во второй постановке состав комплекса заданий W_k и их характеристики становятся известными только в момент τ_k. Тогда планировать выполнение работ W_k возможно только после поступления соответствующего запроса, который поступает в момент τ_k, т.е. в режиме реального времени.

В обеих постановках требуется определить, существует ли допустимое расписание для всего комплекса работ:

$$W = \bigcup_{k=1}^{K} W_k$$

(т.е. расписание, при котором каждое задание выполняется в своем директивном интервале) и построить его в случае положительного ответа. Предполагается, что временем работы самого алгоритма построения расписания в обоих случаях можно пренебречь.

Решение поставленной задачи основано на построении сетевой потоковой модели и поиске максимального потока. Поэтому в следующем разделе дается описание одного из наиболее эффективных потоковых алгоритмов, модификация которого будет использована для построения расписания.

2. Краткое описание полиномиального алгоритма поиска максимального потока в сети. Дана сеть $G = (V, A)$, V — множество вершин, u — источник, v — сток, $u, v \in V$, A — множество ориентированных дуг, $U(a,b)$ — пропускная способность дуги $(a,b) \in A$. В [20] предложен следующий полиномиальный алгоритм поиска максимального потока в сети G.

Шаг 1. Выбрать в качестве начального нулевой поток f, т.е. положить $f(a,b) = 0$ для всех $(a,b) \in A$.

Шаг 2. Построить остаточную сеть $G(f) = (V, A(f))$:
если $f(a,b) < U(a,b)$, то включить в $A(f)$ дугу (a,b) с пропускной способностью $U(a,b) - f(a,b)$;
если $f(a,b) > 0$, то включить в $A(f)$ дугу (b,a) с пропускной способностью $f(a,b)$.

Шаг 3. Если в сети $G(f)$ не существует прямого пути из u в v, то f — максимальный поток; алгоритм завершен. В противном случае перейти на шаг 4.

Шаг 4. Построить слоистую сеть $G^*(f) = \left(V^*(f),\ A^*(f)\right)$. Она содержит все кратчайшие ориентированные пути из u в v.

Шаг 5. Найти тупиковый поток g в сети $G^*(f)$. (Тупиковый поток — это поток, относительно которого нет прямого увеличивающего пути.)

Шаг 5.1. Определить узел $a_0 \in V^*(f)$ с минимальной пропускной способностью. (Пропускная способность узла — это минимум из максимальной величины потока, который может войти в этот узел, и максимальной величины потока, который может выйти из него.)

Шаг 5.2. “Протолкнуть” из узла a_0 влево вплоть до источника u и вправо вплоть до стока v максимально возможный поток. Полученные при этом потоки по дугам определяют поток g.

Шаг 5.3. Удалить из сети $G^*(f)$ узел a_0 и все другие узлы с нулевой остаточной пропускной способностью, инцидентные этим узлам дуги, все полностью насыщенные дуги, образовавшиеся “висячие” узлы (если таковые имеются) и все инцидентные им дуги.

Шаг 5.4. Если существует путь из u и v в оставшейся части сети $G^*(f)$, то перейти на шаг 5.1. В противном случае тупиковый поток g в сети $G^*(f)$ построен.

Шаг 6. Произвести коррекцию потока f в сети G:
если дуге $(a,b) \in A$ соответствует дуга $(a,b) \in A^*(f)$, то положить $f(a,b) = f(a,b) + g(a,b)$; если дуге $(a,b) \in A$ соответствует дуга $(b,a) \in A^*(f)$, то положить $f(a,b) = f(a,b) - g(b,a)$. Перейти на шаг 2.

Вычислительная сложность описанного алгоритма составляет $O\left(|V|^3\right)$.

3. Краткое описание алгоритма В.С. Танаева построения допустимого расписания. Дальнейшее исследование поставленной задачи основано на использовании алгоритма В.С. Танаева построения допустимого многопроцессорного расписания с прерываниями и переключениями [1]. Приведем краткое описание этого алгоритма для сформулированной в разд. 1 задачи при $K = 1$. Будем предполагать, что в этом случае множество работ $W = \{w_1, w_2, \ldots, w_n\}$, их длительности t_i и директивные интервалы $[b_i, c_i]$, m — число идентичных процессоров.

Пусть $y_0 < y_1 < \ldots < y_p$ — все различные величины b_i, c_i, $i = \overline{1,n}$, $I_j = [y_{j-1}, y_j]$, $\delta_j = y_j - y_{j-1}$, $j = \overline{1,p}$. Строится потоковая сеть $G = (V, A)$ (см. рисунок), где $V = \{u, I_j, w_i, v\}$ — множество вершин, u — источник, v — сток, $A = \{(u, I_j), (I_j, w_i), (w_i, v)\}$ — множество ориентированных дуг. Дуга (I_j, w_i) вводится в сеть, если $I_j \in [b_i, c_i]$. Отметим, что при всех j и i либо $I_j \in [b_i, c_i]$, либо $I_j \cap [b_i, c_i] = \varnothing$. Пропускные способности $U \in$ дуг определяются следующим образом: $U(u, I_j) = m\delta_j$, $U(I_j, w_i) = \delta_j$, $U(w_i, v) = t_i$.

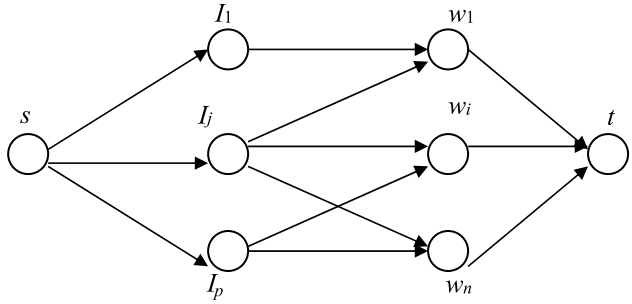

Рисунок. Потоковая сеть G для поиска допустимого расписания

В [1] доказано, что допустимое расписание существует в том и только том случае, когда максимальный поток f в сети G насыщает все выходные дуги (w_i, v), т.е. когда $f(w_i, v) = t_i$ при всех $i = \overline{1,n}$. Величина $f(I_j, w_i)$ равна процессорному времени, выделяемому работе w_i в интервале I_j. Для построения допустимого расписания следует в каждом интервале I_j применить алгоритм упаковки [1], вычислительная сложность которого составляет $O(n)$. Для поиска максимального потока в сети G можно использовать полиномиальный алгоритм, описанный в разд. 2. Поскольку $p \le 2n$, то вычислительная сложность алгоритма В.С. Танаева в этом случае составляет $O(n^3)$.

4. Построение расписания для случая, когда информация о множествах W_k, $k = \overline{1,K}$, поступает до момента τ_1. Рассмотрим случай, когда заранее (т.е. до момента времени τ_1) известны состав всех множеств W_k и характеристики входящих в них работ. В этом случае еще до поступления запросов на выполнение работ W_k в моменты времени τ_k (т.е. до момента времени τ_1) можно составить расписание для всего комплекса заданий W.

Пусть y_k^j — это все различные величины τ_k, b_k^i и c_k^i, $k = \overline{1,K}$, $i = \overline{1,r_k}$. Далее, так же как в разд. 3, определяются интервалы I_k^j и потоковая сеть $G_k = (V_k, A_k)$, где $V_k = \left\{u, v, I_k^j, w_k^i,\ j = \overline{1,p_k}\right\}$ — множество узлов, u — источник, v — сток, $A_k = \left\{\left(u, I_k^j\right), \left(I_k^j, w_k^i\right), \left(w_k^i, v\right)\right\}$ — множество ориентированных дуг. Дуга $\left(I_k^j, w_k^i\right)$ вводится в сеть G_k, если $I_k^j \in [b_k^i, c_k^i]$.

Далее, для решения вопроса о существовании и построении допустимого расписания может быть использован алгоритм, аналогичный тому, который описан в разд. 3. Вычислительная сложность алгоритма составляет

$$O\left(\left(\sum_{k=1}^{K} r_k\right)^3\right).$$

5. Построение расписания для случая, когда информация о множестве W_k поступает в момент τ_k, $k = \overline{1,K}$. Рассматривается случай, когда моменты τ_k поступления запросов на выполнение работ W_k известны заранее, а состав каждого множества W_k и характеристики входящих в него заданий становятся известными только в момент τ_k. Сначала рассмотрим задачу построения расписания для W_1.

5.1. Построение сетевой модели и допустимого расписания для W_1. Пусть $y_1^0 < y_1^1 < \ldots < y_1^{p_1}$ — все различные величины среди τ_1, τ_2, b_1^i, c_1^i, $i = \overline{1,r_1}$, принадлежащие интервалу $[\tau_1, \tau_2]$. Определим интервалы $I_1^j = \left[y_1^{j-1}, y_1^j\right]$, $j = \overline{1,p_1}$, и сеть $G_1 = (V_1, A_1)$, где $V_1 = \left\{u, v, I_1^j, w_1^i\right\}$, а множество ориентированных дуг A_1 и их пропускные способности $U_1(i,j)$ определяются по аналогии с тем, как это сделано в разд. 3 при $m = m_1$, $n = r_1$.

При нахождении максимального потока f_1 в сети G_1 используется следующая модификация алгоритма, описанного в разд. 2. При проталкивании потока из вершины I_1^j вправо (шаг 5.2 разд. 2) в первую очередь следует использовать дуги (I_1^j, w_1^i) для работ w_1^i, у которых $c_1^i \le \tau_2$.

Это объясняется тем, что такие задания могут выполняться только в интервале $[\tau_1, \tau_2]$, а работам с директивным сроком, большим τ_2, процессорное время может выделяться и после момента τ_2.

Величина потока $f_1(I_1^j, w_1^i)$ равна объему процессорного времени, выделенному работе w_1^i в интервале I_1^j. Расписание в интервале I_1^j строится с помощью алгоритма упаковки.

Если хотя бы для одной работы w_1^i с директивным сроком $c_1^i \le \tau_2$ оказалось, что $f_1\left(w_1^i, v\right) < t_1^i$ (т.е. дуга $\left(w_1^i, v\right)$ насыщена не полностью), то допустимого расписания для W_1, и, следовательно, для всего комплекса заданий W не существует. В случае когда $f_1\left(w_1^i, v\right) = t_1^i$ для всех работ w_1^i с директивным сроком $c_1^i \le \tau_2$, построение допустимого расписания продолжится в момент τ_2 для незавершенных работ из W_1 (если таковые имеются) и вновь поступивших работ из W_2.

5.2. Построение сетевой модели и допустимого расписания для $W_1 \cup \ldots \cup W_k, k = \overline{2,K}$. Предположим, что построена потоковая сеть $G_{k-1}, k = \overline{2,K}$, и в ней найден максимальный поток f_{k-1}. Если хотя бы для одной работы $w_r^i, r = \overline{2,k-1}$, для которой $c_r^i \le \tau_k$, после нахождения максимального потока f_{k-1} в сети G_{k-1} оказалось, что $f_{k-1}\left(w_r^i, v\right) < t_r^i$, т.е. дуга $\left(w_r^i, v\right)$ насыщена не полностью, то допустимого расписания для $W_1 \cup \ldots \cup W_{k-1}$, и, следовательно, для всего комплекса W не существует. Если же $f_{k-1}\left(w_r^i, v\right) = t_r^i$ для всех дуг $\left(w_r^i, v\right)$, $r = \overline{1,k-1}$, для которых $c_r^i \le \tau_{k-1}$, то построение допустимого расписания для W продолжается в момент τ_k для незавершенных работ из множества $W_1 \cup \ldots \cup W_{k-1}$ (если таковые имеются) и вновь поступивших заданий из W_k.

Пусть $y_k^0 < y_k^1 < \ldots < y_k^{p_k}$ — все различные величины τ_k, τ_{k+1}, b_k^i, c_k^i, $i = \overline{1,r_k}$. Определим интервалы $I_k^j = \left[y_k^{j-1}, y_k^j\right]$ и пусть $\delta_k^j = y_k^j - y_k^{j-1}$. Из сети G_{k-1} удалим следующие элементы:
узлы I_{k-1}^j, $j = \overline{1,p_{k-1}}$, и все инцидентные им дуги $\left(u, I_{k-1}^j\right)$, $\left(I_{k-1}^j, w_{k-1}^i\right)$, $j = \overline{1,p_k}$, $i = \overline{1,r_{k-1}}$;
узлы w_z^i, для которых

$$f_{k-1}\left(w_z^i, v\right) = t_z^i, \tag{5.1}$$

и соответствующие дуги $\left(w_z^i, v\right)$, $z = \overline{1,k-1}$, $i = \overline{1,r_z}$ (поскольку выполнение условия (5.1) означает завершение работы w_z^i).

Далее, длительности каждой оставшейся работы w_z^i уменьшаются на величину потока $f_{k-1}\left(w_z^i, v\right)$ по дуге $\left(w_z^i, v\right)$. Сеть G_k строится из оставшейся не удаленной части сети G_{k-1} путем добавления к ней узлов I_k^j, w_k^i и дуг $\left(I_k^j, w_z^i\right)$, $j = \overline{1,p_k}$, $i = \overline{1,r_z}$, $z = \overline{1,k}$, $\left(w_k^i, v\right)$, $i = \overline{1,r_k}$, по аналогии с тем, как это описано в разд. 3 при $m = m_k$, $n = n_k +$ (*число оставшихся вершин в* G_{k-1}). В сети G_k находится максимальный поток f_k. По аналогии с разд. 5.1 при проталкивании потока из вершины I_k^j вправо (шаг 5.2 разд. 2) в первую очередь следует использовать дуги $\left(I_k^j, w_k^i\right)$ для работ w_k^i, у которых $c_k^j \le \tau_{k+1}$. Величина потока $f_k\left(I_k^j, w_k^i\right)$ равна объему процессорного времени, выделяемому работе w_k^i в интервале I_k^j. Расписание выполнения работ в интервале I_k^j находится с помощью алгоритма упаковки [1].

Вычислительная сложность предложенного алгоритма составляет

$$O\left(K\left(\sum_{k=1}^{K} r_k\right)^3\right).$$

6. Распределение неоднородного комплекса ресурсов. Предполагается, что при выполнении работ помимо процессоров используется $\overline{1,K}$ невозобновляемый ресурс, количество которого в интервале $[\tau_k, \tau_{k+1}]$ составляет S_k, $k = \overline{1,K}$. Работе w_k^i невозобновляемый ресурс может быть выделен только в момент времени τ_k. Если его объем составляет s_k^i, то длительность выполнения задания w_k^i равна

$$t_k^i = \varphi_k^i\left(s_k^i\right), \tag{6.1}$$

где

$$s_k^i \in \left[0,\ \overline{s}_k^i\right], \tag{6.2}$$

$$\sum_{i=1}^{r_k} s_k^i \le S_k. \tag{6.3}$$

Здесь φ_k^i — строго убывающая на интервале $\left[0, \overline{s}_k^i\right]$ функция, принимающая положительные значения, $\overline{s}_k^i$ — заданные величины, $i = \overline{1,r_k}$.

В этих предположениях для решения задачи, поставленной в разд. 1, нам понадобится обобщение алгоритма В.С. Танаева.

6.1. Обобщение алгоритма В.С. Танаева на случай наличия невозобновляемого ресурса. Будем использовать обозначения, введенные в разд. 3. Кроме того, символы t_k^i, s_k^i, $\overline{s}_k^i$, S_k, φ_k^i, W_k, w_k^i заменим на t_i, s_i, s_i, S, φ_i, W, w_i соответственно. В этом случае ограничения (6.1)—(6.3) записываются следующим образом:

$$t_i = \varphi_i\left(s_i\right), \tag{6.4}$$

$$s_i \in \left[0, \overline{s_i}\right], \tag{6.5}$$

$$\sum_{i=1}^{n} s_i \le S. \tag{6.6}$$

Докажем следующие утверждения.

Лемма 1. Допустимое расписание выполнения множества работ W в задаче без невозобновляемого ресурса (без ограничений (6.4)—(6.6)), при котором заданию w_i предоставляется процессорное время в объеме f_i, существует в том и только том случае, когда в сети G существует поток $f(a,b),(a,b) \in A$, такой, что выполнены равенства

$$f\left(w_i, v\right) = f_i \tag{6.7}$$

при всех $i = \overline{1,n}$.

Доказательство следует из [1]. Пусть $\varphi_i^{-1}\left(s_i\right)$ — функция, обратная по отношению к $\varphi_i\left(s_i\right)$.

Лемма 2. Допустимое расписание выполнения множества работ W в задаче с невозобновляемым ресурсом (с учетом ограничений (6.4)—(6.6)), при котором заданию w_i предоставляется процессорное время в объеме f_i, существует в том и только том случае, когда в сети G существует поток $f(a,b),(a,b) \in A$, такой, что справедливы равенства (6.7) и неравенства

$$\sum_{i=1}^{n} \varphi_i^{-1}(f_i) \le S, \tag{6.8}$$

$$\varphi_i^{-1}\left(f_i\right) \le \overline{s_i},\ i = \overline{1,n}. \tag{6.9}$$

Доказательство. 1. Пусть в сети G существует поток f, для которого справедливы соотношения (6.7)—(6.9). В этом случае из (6.8) следует, что каждой работе w_i может быть выделен невозобновляемый ресурс в объеме $s_i = \varphi_i^{-1}\left(f_i\right)$. При этом будет справедливо неравенство (6.6),

а в силу (6.4) длительность выполнения работы w_i (т.е. требуемое процессорное время) составит $\varphi_i\left(\varphi_i^{-1}\left(f_i\right)\right) = f_i$. Поскольку $\varphi_i^{-1}\left(f_i\right) = s_i$, то из (6.9) вытекает (6.5). Тогда из леммы 1 следует существование допустимого расписания для W с учетом ограничений (6.4)—(6.6).

2. Пусть теперь существует допустимое расписание выполнения работ W в задаче с невозобновляемом ресурсом (с учетом ограничений (6.4)—(6.6)), при котором заданию w_i выделяется процессорное время в объеме f_i. Пусть при этом работе w_i выделен невозобновляемый ресурс в объеме s_i. Тогда в силу (6.4) $f_i = \varphi_i\left(s_i\right)$ или $s_i = \varphi_i^{-1}\left(f_i\right)$. В этом случае из (6.6) и (6.5) следуют неравенства (6.8) и (6.9) соответственно. Теперь доказательство вытекает из леммы 1. Лемма доказана.

Из (6.7), (6.9) следует, что при построении допустимого расписания в задаче с невозобновляемым ресурсом нужно искать поток f в сети G, для которого $f\left(w_i, v\right) = f_i \le \varphi\left(\overline{s_i}\right)$ при всех $i = \overline{1,n}$. С учетом того, что функции $\varphi_i^{-1}\left(f_i\right)$ строго убывают, величины f_i необходимо максимизировать. Поэтому для решения этой задачи предлагается следующий алгоритм.

В сети G сначала определяется вершина w_{i_1}, для которой величина максимального потока g из u в w_i является наибольшей среди всех вершин w_i, $i = \overline{1,n}$. Далее, номер i_1 включается в множество N и пропускные способности всех дуг $(a,b) \in A$ сети G уменьшаются на величину $g(a,b)$. После чего данная процедура повторяется для вершин w_i, $i = \overline{1,n}$, iN. Далее, выполняется проверка условия (6.8).

Алгоритм 1.

Шаг 1. В сети G положить $U\left(w_i, v\right) = \varphi_i\left(\overline{s_i}\right)$, $N = \varnothing$.

Шаг 2. Для $i_0 = \overline{1,n}, i_0 \in N$, выполнять шаги 3—5.

Шаг 3. Удалить из сети G все дуги $\left(w_i, v\right)$, $i = \overline{1,n}$, $i \ne i_0$.

Шаг 4. Найти максимальный поток g в сети G и пусть g_{i_0} — его величина.

Шаг 5. Включить в сеть G дуги, удаленные на шаге 3.

Шаг 6. Пусть $\min\limits_{i_0 = \overline{1,n},\ i_0 \in N} \varphi_{i_0}^{-1}\left(g_{i_0}\right) = g_{i_1}$. Включить i_1 в N. Положить $f_{i_1} = g\left(w_{i_1}, v\right) = g_{i1}$. Пропускные способности $U(a,b)$ всех дуг $(a,b) \in A$ сети G уменьшить на величину $g(a,b)$.

Шаг 7. Если $|N| < n$, то перейти на шаг 2. Если $|N| = n$, то перейти на шаг 8.

Шаг 8. Если выполнено неравенство (6.8), то допустимое расписание существует. При этом работе w_i выделяется невозобновляемый ресурс в объеме $\varphi_i^{-1}\left(f_i\right)$, $i = \overline{1,n}$. Длительность выполнения работы w_i составляет f_i. Расписание строится так, как описано в разд. 3. Если неравенство (6.8) не выполнено, то допустимого расписания не существует. Алгоритм завершен.

Вычислительная сложность алгоритма 1 составляет $O\left(n^5\right)$.

6.2. Обобщение исходной задачи на случай наличия нево-зобновляемого ресурса. Перейдем теперь к задаче, сформулированной в разд. 1, с дополнительными ограничениями (6.1)—(6.3), связанными с распределением невозобновляемого ресурса. Вновь вернемся к обозначениям t_k^l, s_k^l, $\overline{s}_k^l$, S_k, φ_k^l, W_k, w_k^l, которые ранее были заменены на t_i, s_i, s_i, S, φ_i, W, w_i соответственно.

С учетом исследований, проведенных в разд. 5.1, 5.2 и 6.1, предлагается следующий алгоритм решения исходной задачи, сформулированной в разд. 1, для случая наличия невозобновляемого ресурса.

Алгоритм 2.

Шаг 1. Положить $k = 1$.

Шаг 2. Построить сеть G_k (см. разд. 5.1, 5.2).

Шаг 3. Из сети G_k удалить узлы w_m^i, соответствующие работам с директивным сроком $c_m^i > \tau_{k+1}$, и инцидентные им дуги.

Шаг 4. К полученной сети применить алгоритм 1. Если на шаге 8 алгоритма 1 выяснилось, что допустимого расписания не существует, то алгоритм 2 завершен, решения не существует. В противном случае перейти на шаг 5.

Шаг 5. Включить в сеть G_k удаленные на шаге 3 узлы и дуги. Положить $k = k + 1$. Если $k \le K$, то перейти на шаг 1. Если $k > K$, то решение построено. Алгоритм завершен.

Вычислительная сложность алгоритма 2 составляет

$$O\left(K\left(\sum_{k=1}^{K} r_k\right)^5\right).$$

Заключение. Исследована задача составления многопроцессорного допустимого расписания для совокупности комплексов работ, запросы на выполнение которых поступают в заданные моменты времени. Состав каждого комплекса и характеристики входящих в него работ становятся известными в момент поступления запроса. При выполнении заданий допускаются прерывания и переключения с одного процессора на другой. Исследованы постановки с невозобновляемым ресурсом и без него. Разработан полиномиальный алгоритм решения задачи, основанный на построении сетевой потоковой модели и поиске максимального потока.

СПИСОК ЛИТЕРАТУРЫ

1. *Танаев В. С.*, *Гордон В. С.*, *Шафранский Я. М.* Теория расписаний. Одностадийные системы. М.: Наука, 1984.
2. *Brucker P.* Scheduling Algorithms. Heidelberg: Springer, 2007.
3. *Лазарев А. А.* Теория расписаний. Оценка абсолютной погрешности и схема приближенного решения задач теории расписаний. М.: МФТИ, 2008.
4. *Горский М. А.*, *Мищенко А. В.*, *Нестерович Л. Г.*, *Халиков М. А.* Некоторые модификации целочисленных оптимизационных задач с учетом неопределенности и риска // Изв. РАН. ТиСУ. 2022. № 5. С. 106—117.
5. *Мищенко А. В.*, *Кошелев П. С.* Оптимизация управления работами логистического проекта в условиях неопределенности // Изв. РАН. ТиСУ. 2021. № 4. С. 123—134.
6. *Глонина А. Б.*, *Балашов В. В.* О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов // Моделирование и анализ информационных систем. 2018. Т. 25. № 2. С. 174—192.
7. *Глонина А. Б.* Обобщенная модель функционирования модульных вычислительных систем реального времени для проверки допустимости конфигураций таких систем // Вестн. ЮУрГУ. Сер. Вычисл. математика и информатика. 2017. Т. 6. № 4. С. 43—59.
8. *Глонина А. Б.* Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем // Вестн. МГУ. Сер. 15. Вычисл. математика и кибернетика. 2020. № 3. С. 16—29.
8. *Алифанов Д. В.*, *Лебедев В. Н.*, *Цурков В. И.* Оптимизация расписаний с логическими условиями предшествования // Изв. РАН. ТиСУ. 2009. № 6. С. 88—93.
9. *Миронов А. А.*, *Цурков В. И.* Минимакс в моделях транспортного типа с интегральными ограничениями // Изв. РАН. ТиСУ. 2003. № 4. С. 69—81.
10. *Миронов А. А.*, *Цурков В. И.* Минимакс при нелинейных транспортных ограничениях // ДАН. 2001. Т. 381. № 3. С. 305—308.
11. *Филлипс Д.*, *Гарсиа-Диас А.* Методы анализа сетей. М.: Мир, 1984.
12. *Давыдов Э. Г.* Исследование операций. М.: Высш. шк., 1990.
13. *Фуругян М. Г.* Планирование вычислений в многопроцессорных системах с несколькими типами дополнительных ресурсов и произвольными процессорами // Вестн. МГУ. Сер. 15. Вычисл. математика и кибернетика. 1917. № 3. С. 38—45.
14. *Yao X.*, *Almatooq N.*, *Askin R. G.*, *Gruber G.* Capacity Planning and Production Scheduling Integration: Improving Operational Efficiency Via Detailed Modelling // Intern. J. Production Research. Published Online. 2022. V. 60. No. 1.
15. *Missbauer H.*, *Uzsoy R.* Order Release in Production Planning and Control Systems: Challenges and Opportunities // Intern. J. Production Research. 2022. V. 60. No. 1.
16. *Wang Y.*, *Geunes J.*, *Nie X.* Optimising Inventory Placement in a Two-echelon Distribution System with Fulfillment-time-dependent Demand // Intern. J. Production Research. 2022. V. 60. No. 1.
17. *Gorman M.F.*, *Conway D.* G. ATtutorial of Integrating Duality and Branch and Bound in Earliness-tardiness Scheduling with Idle Insertion Time Problems // Intern. J. Production Research. 2018. V. 56. No. 1-2.
18. *Graves S. C.* How to Think About Planned Lead Times // Intern. J. Production Research. 2022. V. 60. No. 1.
19. *Thomasson O.*, *Battarra M.*, *Erdoğan G.*, *Laporte G.* Scheduling Twin Robots in a Palletising Problem // Intern. J. Production Research. 2018. V. 56. No. 1-2.
20. *Кормен Т.*, *Лейзерсон Ч.*, *Ривест Р.*, *Штайн К.* Алгоритмы: построение и анализ. М.: Вильямс, 2005.