УДК 519.86 doi: 10.15827/0236-235X.149.055-064 2025. Т. 38. № 1. С. 55–64

Планирование вычислений в системах реального времени: эффективные алгоритмы построения оптимальных расписаний

Д.А. Кононов [1], М.Г. Фуругян [2]✉

[1] Институт проблем управления им. В.А. Трапезникова РАН,
г. Москва, 117997, Россия
[2] Федеральный исследовательский центр «Информатика и управление» РАН,
г. Москва, 119333, Россия

Ссылка для цитирования
Кононов Д.А., Фуругян М.Г. Планирование вычислений в системах реального времени: эффективные алгоритмы построения оптимальных расписаний // Программные продукты и системы. 2025. Т. 38. № 1. С. 55–64. doi: 10.15827/0236-235X.149.055-064
Информация о статье
Группа специальностей ВАК: 1.2.3
Поступила в редакцию: 02.08.2024 После доработки: 13.08.2024 Принята к публикации: 23.08.2024

Аннотация. В статье рассматриваются вопросы, связанные с разработкой одного из основных блоков вычислительной системы реального времени – блока планирования вычислений. Предлагаются алгоритмы построения оптимальных расписаний для различных случаев в зависимости от числа процессоров и характеристик работ и ресурсов вычислительной системы. Для однопроцессорного случая с прерываниями и директивными интервалами усовершенствован алгоритм относительной срочности путем использования кучи для хранения данных. Это способствовало понижению вычислительной сложности алгоритма. Разработан алгоритм для задачи с частичным порядком выполнения работ, основанный на предварительной коррекции моментов готовности и директивных сроков и на сведении исходной задачи к задаче без отношений предшествования. Для многопроцессорного случая с прерываниями и директивными интервалами предложен приближенный алгоритм, основанный на обобщении однопроцессорного алгоритма относительной срочности на случай нескольких процессоров. Проведен сравнительный анализ с точным потоковым алгоритмом. Доказано, что в случае учета временных издержек на прерывания и переключения задача является NP-трудной. Для многопроцессорного случая без прерываний и переключений с общим директивным интервалом для всех работ и идентичными процессорами разработан псевдополиномиальный алгоритм, основанный на ограниченном переборе вариантов. Создан приближенный алгоритм для системы с возобновляемыми и невозобновляемыми ресурсами, а также для комплекса со смешанным набором работ (как непрерываемых, так и допускающих прерывания и переключения). Алгоритм основан на сетевом моделировании и сведении исследуемой задачи к поиску потока с определенными свойствами в специальной сети.
Ключевые слова: вычислительная система реального времени, однопроцессорная и многопроцессорная системы, потоковая сеть, допустимое расписание, возобновляемые и невозобновляемые ресурсы

Введение. Вычислительные системы реального времени стали широко внедряться в различные сферы деятельности человека с конца 70-х гг. прошлого века. Они находят применение в тех областях, где на проведение вычислений отводится строго ограниченное время. В частности, при проектировании и эксплуатации сложных технических объектов (самолеты, электростанции, атомные реакторы) это время может составлять доли секунды. В ряде случаев на вычисления может отводиться существенно большее время, например, при обработке информации экономического и экологического характера.

Известны два основных подхода к разработке вычислительных систем реального времени. Первый подход основан на предварительном расчете расписания выполнения вычислений. Так, авторами [1–3] разработана методика построения допустимых расписаний с директивными сроками в многоядерной вычислительной системе реального времени. Согласно этой методике, сначала строится временная диаграмма работы системы, а затем с ее помощью проверяется выполнение каждого задания в своем директивном интервале. Кроме того, разработана имитационная модель системы, основанная на использовании сетей Петри и обобщенных конечных автоматов с остановкой таймера.

Авторы данной статьи в разное время принимали участие в проектах по созданию вычислительных систем реального времени: для обработки информации при проведении летных испытаний, для функционирования системы противовоздушной и космической обороны, при разработке газовых месторождений и строительных работах. При этом также использовался подход, основанный на предварительном расчете расписания выполнения вычислений,

как это было сделано при разработке системы автоматизации программирования вычислительных систем реального времени, предназначенных для обработки циклически поступающей информации. На вход системы поступает программа пользователя, в которой описаны прикладные модули, предназначенные для вычисления, их длительности, входные и выходные параметры, частоты обращения к ним, длительности их выполнения и другие параметры. Блок генерации кодов переводит информацию, содержащуюся в этой программе, в таблицы, удобные для дальнейшей обработки. Далее работает генератор сетевой модели, который строит ориентированную сеть. Узлами сети являются прикладные модули, а дуги указывают отношения частичного порядка их выполнения. Затем подключается генератор расписаний, который, получая на вход эту сетевую модель, определяет, существует ли допустимое расписание выполнения прикладных модулей, и строит его в случае положительного ответа. Управляющая программа запускает в реальном времени прикладные модули согласно построенному расписанию. Одним из основных блоков этой системы является генератор расписаний.

При втором подходе планирование вычислений осуществляется в режиме реального времени. Так, в [4] исследована задача планирования комплекса работ, требования на выполнение которых поступают в известные моменты времени. Однако характеристики работ, такие как длительность и директивные сроки, становятся известными в момент поступления каждого запроса. В этом случае возникает необходимость составлять расписание выполнения вычислений в режиме реального времени. Авторами разработан полиномиальный алгоритм построения допустимого расписания для данного случая.

Таким образом, оба подхода к созданию вычислительных систем реального времени требуют эффективных алгоритмов составления расписаний. С появлением многопроцессорной и многоядерной вычислительной техники проблемы планирования вычислений стали еще более актуальными. Этому посвящено немало публикаций. Так, в [5] приводится большое число алгоритмов как для однопроцессорных, так и для многопроцессорных систем. В частности, впервые решена задача построения расписания для прерываемых работ с директивными интервалами в системе с идентичными процессорами путем сведения ее к потоковой задаче. Рассмотрены задачи как с независимыми заданиями, так и с множеством работ с частичным порядком их выполнения. В [6] классифицированы задачи по теории расписаний, исследуются некоторые задачи дискретной оптимизации, метод ветвей и границ. В работах [5, 6] проведен анализ вычислительной сложности задач и алгоритмов.

В работах [7–9] рассматривается ряд экономических задач планирования. Предполагается, что некоторые параметры (длительность выполнения работ, объем ресурсов) не являются фиксированными, а могут принимать значения из заданных интервалов либо представляют собой случайные величины. Для решения этих задач авторы используют метод ветвей и границ. В [10] на основе понятия расстояния между задачами разработаны методы решения ряда NP-трудных задач для критерия минимизации максимального запаздывания, а также для задач на быстродействие.

В работе [11] рассмотрены системы реального времени с интегрированной модульной архитектурой. Авторы предлагают алгоритмы составления расписаний, основанные на построении транспортной сети и нахождении в ней максимального потока. Создание алгоритма решения задачи на быстродействие в многопроцессорной системе при ограничении на количество передач данных между процессорами описано в [12]. Алгоритм основан на использовании метода имитации отжига.

В [13] описана процедура построения расписания, состоящая из двух шагов. На первом шаге с помощью эвристического алгоритма, основанного на методе ветвей и границ, определяется последовательность выполнения работ. На втором в построенный график добавляются интервалы простоя с учетом директивных сроков начала и окончания операций. Эта задача сводится к задаче линейного программирования. В [14] рассмотрены проблемы, связанные с установлением директивных сроков выполнения работ и допустимых отклонений от этих сроков. Предложена модель, показывающая связь директивных сроков и стохастической изменчивости объемов ресурсов, необходимых для выполнения работ. В [15] решается задача оптимизации работы и маршрутов двух роботов, которые доставляют продукты в определенные места и должны вернуться в исходное положение. Рассмотрена задача на быстродействие и доказана ее NP-трудность. Для ее решения используются методы целочисленного линейного программирования, а также генети-

ческий алгоритм. Для оценки качества решений применяется динамическое программирование.

В настоящей статье предлагаются алгоритмы составления однопроцессорных и многопроцессорных расписаний с директивными сроками для работ, допускающих прерывания и переключения с одного процессора на другой, а также для непрерываемых работ. Особенность разработки в том, что исследованы задачи, совокупность работ которых включает не только прерываемые, но и непрерываемые задания, а комплекс ресурсов состоит из возобновляемых и невозобновляемых ресурсов. При этом показано, как выбор подходящей структуры данных позволяет сократить вычислительную сложность алгоритмов.

Составление однопроцессорных и многопроцессорных расписаний с прерываниями и директивными сроками

- *Построение однопроцессорного расписания*

Имеется совокупность работ (заданий) $W = \{w_1, w_2, \ldots, w_n\}$, которые должны быть выполнены с помощью m идентичных процессоров. Известны длительности t_i выполнения работ w_i и их директивные интервалы $[b_i; f_i]$, $t_i \le f_i - b_i$ (работа w_i не может быть начата раньше момента ее готовности b_i и должна быть завершена не позднее ее директивного срока f_i). При выполнении работ допускаются прерывания и переключения с одного процессора на другой. Предполагается, что они не требуют временных затрат. Не допускаются параллельное выполнение одного задания несколькими процессорами и одновременное выполнение нескольких работ одним процессором. Требуется определить, существует ли допустимое расписание выполнения работ, и найти его, если оно существует. Допустимое расписание – это такое расписание, при котором каждая работа выполняется в своем директивном интервале.

Алгоритм относительной срочности. Для однопроцессорных систем ($m = 1$) известен алгоритм [16], который выглядит следующим образом.

Алгоритм 1.

Пусть $A(t)$ – множество активных работ в момент времени t, то есть таких работ w_i, которые еще не завершены (или не начаты), и при этом $t \in [b_i; f_i]$. Процессор выделяется активной работе w_i с минимальным директивным сроком $f : f_{i_0} = \min\limits_{i:\, w_i \in A(t)} f_i$. Если таких работ несколько, то выбирается любая из них. Работа w_i выполняется до тех пор, пока либо она не завершится, либо не появится активная работа w_{j_0} с меньшим директивным сроком f_{j_0}. В последнем случае выполнение работы w_{i_0} прерывается и процессор передается заданию w_{j_0}.

Поскольку выполнение работ может прерываться только в моменты времени b_i, число прерываний не более n. В отличие от [16] величины f_i для активных работ будем хранить в виде двоичной кучи с минимальным элементом в вершине. В этом случае вычислительная сложность алгоритма 1 составляет $O(n \log n)$. Алгоритм, описанный в [16], не использует двоичную кучу и имеет вычислительную сложность $O(n^2)$. В работе доказано, что данный алгоритм является корректным: если он не находит решение, то оно не существует.

Построение однопроцессорного расписания с отношением частичного порядка на множестве работ. Сформулированную постановку задачи для однопроцессорной системы дополним еще одним условием. Будем предполагать, что на множестве работ W задано отношение частичного порядка в виде ориентированного графа без циклов $G = (W, A)$, где A – множество ориентированных дуг. Если $(w_i, w_j) \in A$, значит, работа w_j может быть начата только после завершения работы w_i. Здесь w_i – непосредственный предшественник работы w_j, а w_j – непосредственный последователь работы w_i. Предлагаемый алгоритм основан на коррекции директивных интервалов с последующим применением алгоритма 1.

В основе коррекции директивных интервалов лежит следующее простое правило: если $(w_i, w_j) \in A$, то момент готовности b_j работы w_j и директивный срок f_i работы w_i пересчитываются по следующим формулам:

$b_j = \max(b_j, b_i + t_i)$, $f_i = \min(f_i, f_j - t_j)$.

Действительно, $b_i + t_i$ – это наиболее ранний возможный срок окончания работы w_i, а $f_j - t_j$ – наиболее поздний допустимый срок завершения работы w_i. Перед выполнением процедуры коррекции множество W необходимо разбить на уровни следующим образом. Пусть W_0 – это множество работ, не имеющих непосредственных предшественников. Если построены множества $W_0, W_1, \ldots, W_{k-1}$, то W_k – это множество работ, имеющих непосредственных предшественников только в множествах $W_0, W_1, \ldots, W_{k-1}$.

Предположим, что построено разбиение множества W на уровни W_0, W_1, …, W_p, $p \le n-1$. Коррекция моментов готовности работ проводится начиная с W_1, затем W_2 и т.д. следующим образом: если $w_j \in W_k$, то принять

$$b_j = \max\left(b_j, \max_{i:(w_i, w_j)\in A}(b_i + t_i)\right), \ k = \overline{1, p}.$$

Коррекция директивных сроков работ проводится начиная с W_{p-1}, затем W_{p-2} и т.д. следующим образом: если $w_i \in W_k$, то принять

$$f_i = \min\left(f_i, \min_{i:(w_i, w_j)\in A}(f_j - t_j)\right), \ k = \overline{p-1, 0}.$$

Вычислительная сложность процедуры коррекции директивных интервалов составляет $O(n^2)$.

После выполнения процедуры коррекции множество W будет обладать следующим свойством: если $(w_i, w_j) \in A$, то $b_i < b_j$, $f_i < f_j$. Следовательно, для поиска допустимого расписания может быть использован алгоритм относительной срочности (алгоритм 1). Таким образом, алгоритм решения поставленной в настоящем разделе задачи следующий.

Алгоритм 2.

Шаг 1. Выполнить процедуру коррекции директивных интервалов.

Шаг 2. Применить алгоритм 1.

Вычислительная сложность алгоритма 2 составляет $O(n^2)$.

Построение многопроцессорных расписаний. Для случая $m \ge 2$ может быть использован потоковый алгоритм [5] (алгоритм 3), основанный на сведении исходной задачи к нахождению максимального потока в ориентированной сети специального вида. Вычислительная сложность алгоритма 3 составляет $O(n^3)$ (если при нахождении максимального потока использовать алгоритм кубической сложности). Алгоритм 3, как и алгоритм 1, является корректным.

Идея алгоритма относительной срочности в том, что процессор всегда занят активной работой с наименьшим директивным сроком и она легко может быть перенесена на многопроцессорный случай ($m \ge 2$).

Алгоритм 4.

Процессоры всегда должны занимать m активных работ с наименьшими директивными сроками (если таковые имеются). Если активных работ меньше m, то все они должны выполняться. Если появляется новая активная работа с директивным сроком, меньшим наибольшего директивного срока из выполняемых работ, то она заменяет последнюю. В этом случае следует иметь две двоичные кучи. Первая содержит директивные сроки работ, которые выполняются в данный момент (их не более m). В вершине этой кучи находится максимальный элемент. Вторая куча содержит директивные сроки активных работ, которые в данный момент не выполняются. В вершине этой кучи находится минимальный элемент. Возникшая новая активная работа включается во вторую кучу. Если при этом она оказывается в вершине кучи и ее директивный срок меньше директивного срока, находящегося в вершине второй кучи, то элементы, находящиеся в вершинах обеих куч, меняются местами. После этого каждый из них становится на свое место в соответствующей куче. Вычислительная сложность алгоритма 4 составляет $O((n \log n) \log m)$. Алгоритм, описанный в [17], является аналогичным обобщением алгоритма 1 на многопроцессорный случай. Однако он не использует двоичные кучи и имеет более высокую вычислительную сложность: $O(n (m \log m + \log n)$.

Поскольку можно считать, что $m << n$, вычислительная сложность алгоритма 4 существенно меньше вычислительной сложности потокового алгоритма 3. Рассмотрим простой пример, показывающий работу алгоритма 4.

Пример. Пусть $n = 3$, $m = 2$, $b_i = 0$, $f_i = 3$, $t_i = 2$ при всех $i = 1, 2, 3$. Пусть R_{ij} – временной интервал, в котором работа w_i выполняется j-м процессором. Тогда алгоритм 4 построит следующее расписание: $R_{11} = [0; 2]$, $R_{22} = [0; 2]$, $R_{13} = [2; 4]$. Это означает, что работа w_3 не успевает завершиться к своему директивному сроку $f_3 = 3$, то есть алгоритм 4 не находит допустимого расписания. Однако на самом деле допустимое расписание существует: $R_{11} = [0; 2]$, $R_{22} = [0; 1]$, $R_{21} = [2; 3]$, $R_{13} = [1; 3]$. В этом случае все работы успевают завершиться к своему директивному сроку $f_i = 3$, $i = 1, 2, 3$.

Следовательно, алгоритм 4 не всегда работает корректно. Как показано в [17], на основании результатов 25 000 численных экспериментов с рандомизированными и плавно варьируемыми переменными можно сделать вывод, что при больших размерностях задачи алгоритм 4 работает в тысячи раз быстрее алгоритма 3. В то же время некорректная работа алгоритма 4 была отмечена не более чем в 3 % случаев. Поэтому можно предложить следующий алгоритм решения задачи: сначала запускается быстрый алгоритм 4; если решение не найдено, запускается алгоритм 3.

Замечание. Если предположить, что прерывания и переключения с одного процессора на другой требуют временных затрат, то задача становится NP-трудной. Покажем, что в этом

случае к рассматриваемой задаче полиномиально сводится известная NP-полная задача о разбиении (имеются n натуральных чисел a_1, a_2, …, a_n, и пусть $B = \sum_{i=1}^{n} a_i$ – четное число; можно ли разбить это множество на два непересекающихся подмножества с одинаковой суммой элементов, то есть существует ли такое подмножество $\overline{N} \subseteq N = \{1, 2, \ldots, n\}$, что $\sum_{i \in \overline{N}} a_i = \sum_{i \in N \setminus \overline{N}} a_i = B/2$) [18]. Если в рассматриваемой задаче о построении расписания принять $m = 2$, $b_i = 0$, $f_i = B/2$, $i = \overline{1, n}$, то допустимое расписание будет существовать в том и только том случае, когда в задаче о разбиении ответ положительный.

Составление многопроцессорных расписаний без прерываний с общим директивным интервалом

Предположим, что работы не допускают прерываний и переключений с одного процессора на другой, длительности t_i принимают натуральные значения, а директивные интервалы всех работ совпадают: $b_i = 0$, $f_i = F$, $t_i \le F$. Известно, что такая задача является NP-трудной в сильном смысле [18]. Однако при фиксированном числе процессоров m существует псевдополиномиальный алгоритм.

Пусть $X = \{x = (x_1, x_2, \ldots, x_m)\colon 0 \le x_j \le F, j = \overline{1, m}\}$. Здесь x_j – временная загруженность j-го процессора, $j = \overline{1, m}$. Пусть $X_k \subseteq X$, $k = \overline{1, n}$ – непересекающиеся подмножества множества X, которые определяются следующим образом:

$X_1 = \{x \in X\colon x_i = 0$ при $i \ne j$, $x_j = t_j$, $j = \overline{1, m}\}$, то есть X_1 – множество векторов из X, у которых все компоненты, кроме x_j, равны 0, а $x_j = t_j$, $j = \overline{1, m}$;

$X_k = \{x \in X\colon x_j = \overline{x_j} + t_k,\ x_j \le F,\ j = \overline{1, m},\ \overline{x} \in X_{k-1}\}$, $k = \overline{2, n}$, то есть каждый вектор из X_k получается прибавлением величины t_k к j-й компоненте некоторого вектора $\overline{x} \in X_{k-1}$, $j = \overline{1, m}$. Каждому вектору $x \in X_k$ приписывается метка, указывающая, из какого вектора $\overline{x} \in X_{k-1}$ он получен.

Важно отметить: если при построении множества X_k для некоторой компоненты x_j вектора $x \in X_k$ выполняется неравенство $x_j > F$, то вектор x не включается в X_k. К тому же число векторов в X_k не превосходит $(F + 1)^m$. Алгоритм решения задачи (алгоритм 5) заключается в построении множеств X_k, $k = \overline{1, n}$. Если $X_n \ne \varnothing$, то допустимое расписание существует и может быть построено путем следования по меткам, указанным при построении множеств X_k, $k = \overline{2, n}$.

Алгоритм 5.

Шаг 1. Построить множества X_k, $k = \overline{1, n}$.

Шаг 2. Если $X_n \ne \varnothing$, перейти на шаг 3. В противном случае перейти на шаг 5.

Шаг 3. С помощью меток, указанных при построении множеств X_k, $k = \overline{2, n}$, построить последовательность векторов $x^k \in X_k$, $k = \overline{n, 1}$.

Шаг 4. Если x^k и x^{k-1} отличаются j-й компонентой, то работа w_k приписывается процессору j, $k = \overline{n, 1}$. Завершение алгоритма.

Шаг 5. Допустимого расписания не существует. Завершение алгоритма.

Далее работы, приписанные процессору j, $j = \overline{1, m}$, выполняются этим процессором в произвольном порядке. Поскольку число элементов в каждом множестве X_k, $k = \overline{1, n-1}$, не превосходит число элементов в X, то есть величины $(F + 1)^m$, и для каждого из них строится не более m элементов в X_{k+1}, то вычислительная сложность алгоритма 5 составляет $O(mn(F + 1)^m)$, или $O(mnF^m)$. Следовательно, при фиксированном числе процессоров m алгоритм 5 является псевдополиномиальным.

Составление расписаний в системе с неоднородными ресурсами

В работе исследовались задачи с возобновляемыми ресурсами. Это ресурсы, которые могут использоваться многократно, например, процессоры, приборы, машины и т.д. Кроме того, все работы либо допускали прерывания и переключения с одного процессора на другой, либо были непрерываемыми. Рассмотрим задачу планирования работ в системе, где, помимо процессоров, имеются невозобновляемые ресурсы. Эти ресурсы не могут использоваться повторно, например, электроэнергия, финансы, дополнительная память или специализированные устройства, выделяемые конкретному программному модулю и не используемые другими модулями. Кроме того, исследуем задачу, в которой часть работ допускают прерывания и переключения с одного процессора на другой, а часть не допускают.

• *Смешанный комплекс работ, общий директивный интервал*

Рассмотрим комплекс работ $W = W_1 \cup W_2$, где $W_1 = \{w_{11}, w_{12}, \ldots, w_{1n_1}\}$ – работы, допускающие прерывания и переключения с одного процессора на другой, $W_2 = \{w_{21}, w_{22}, \ldots, w_{2n_2}\}$ – непрерываемые работы. Для работ W_1 и W_2 остаются предположения, сделанные ранее. Для всех работ установлен общий директивный интервал $[0; F]$. Для выполнения работ имеются m идентичных процессоров и L типов невозобновляемых ресурсов, объемы которых составляют $R_1, R_2, \ldots, R_L$. Если работе w_{ki} выделен ресурс l-го типа в количестве r_{kil}, $l = \overline{1, L}$, то длительность ее выполнения составляет

$$t_{ki} = t_{ki}^0 - \sum_{l=1}^{L} a_{kil} r_{kil}, \; k = 1, 2, \; i = \overline{1, n_k}, \qquad (1)$$

где t_{ki}^0 – длительность выполнения работы w_{ki} в случае, если невозобновляемые ресурсы ей не выделяются; $t_{ki}^0 \le F$, $k = 1, 2$, $i = \overline{1, n_k}$, $a_{kil} > 0$ – заданные величины. На величины r_{kil} накладываются следующие ограничения:

$$r_{kil}^1 \le r_{kil} \le r_{kil}^2, \; k = 1, 2, \; i = \overline{1, n_k}, \; l = \overline{1, L}, \qquad (2)$$

$$\sum_{k=1}^{2} \sum_{i=1}^{n_k} r_{kil} \le R_l, \quad l = \overline{1, L}, \qquad (3)$$

$$t_{ki}^0 - \sum_{l=1}^{L} a_{kil} r_{kil}^2 > 0, \; k = 1, 2, \; i = \overline{1, n_k}. \qquad (4)$$

Неравенство (2) определяет ограничения на объемы ресурсов, выделяемых каждой работе. Неравенство (3) ограничивает общий объем выделяемых ресурсов каждого типа. Согласно неравенству (4), при выделении работе максимально допустимого объема ресурсов каждого типа ее длительность остается положительной. Распределение ресурсов r_{kil}, $k = 1, 2$, $i = \overline{1, n_k}$, $l = \overline{1, L}$, удовлетворяющее ограничениям (2)–(4), будем называть допустимым. Задача заключается в поиске допустимого распределения ресурсов и допустимого расписания.

Алгоритм 6 решения задачи состоит из трех этапов. На первом этапе выполняется распределение невозобновляемых ресурсов, после чего будут определены длительности выполнения работ. На втором этапе процессоры делятся на две группы: первая – для выполнения работ W_1, вторая – для выполнения работ W_2. На третьем этапе строится расписание, отдельное для W_1 и W_2.

Для каждого $l = \overline{1, L}$ ресурс l-го типа распределяется следующим образом. Определяется $\max\limits_{k, i} a_{kil} = a_{k_0 i_0 l}$, и работе $w_{k_0 i_0}$ выделяется максимально возможное количество ресурса l-го типа (то есть такое количество, при котором не нарушаются ограничения (2), (3)). Оставшаяся неиспользованной часть ресурса l-го типа распределяется среди оставшихся работ по аналогичному правилу. Такое распределение ресурсов позволяет максимально сократить длительности работ. Если для каждого $l = \overline{1, L}$ величины a_{kil} отсортировать по невозрастанию, то вычислительная сложность первого этапа будет составлять $O(Ln \log n)$.

После распределения невозобновляемого ресурса между работами вычисляются их длительности по (1) и величины

$$m_1 = \left[\frac{\sum\limits_{i=1}^{n_1} t_{1i}}{\sum\limits_{i=1}^{n_1} t_{1i} + \sum\limits_{i=1}^{n_2} t_{2i}} \right], \; m_2 = m - m_1. \qquad (5)$$

Работы W_1 будут выполняться первыми m_1 процессорами, а работы W_2 – оставшимися m_2 процессорами. По формуле (5) распределяются процессоры между работами W_1 и W_2 пропорционально суммарной длительности выполнения работ этих множеств. Вычислительная сложность второго этапа составляет $O(n)$, где $n = n_1 + n_2$.

Для построения расписания выполнения работ W_1 и W_2 соответственно могут быть использованы алгоритм упаковки [5], вычислительная сложность которого составляет $O(n_1)$, и рассмотренный псевдополиномиальный алгоритм, вычислительная сложность которого составляет $O\left(m_2 n_2 F^{m_2}\right)$.

• *Прерываемые работы, произвольные директивные интервалы*

Предположим, что множество W однородное: каждая работа $w_i \in W$ допускает прерывания и переключения с одного процессора на другой и имеет директивный интервал $[b_i; f_i]$. Для ресурса l-го типа известен коэффициент a_l, показывающий, на какую величину сократится время выполнения работы, если ей будет выделена единица этого ресурса. К тому же выполнены приведенные далее ограничения, аналогичные условиям (1)–(4), а именно: если работе w_i выделено r_{il} ресурсов l-го типа, то справедливы соотношения:

$$t_i = t_i^0 - \sum_{l=1}^{L} a_l r_{il}, \; i = \overline{1, n}, \qquad (6)$$

$$r_{il} \le r_{il}^2, \; i = \overline{1, n}, \; l = \overline{1, L}, \qquad (7)$$

$$\sum_{i=1}^{n} r_{il} \le R_l,\ l = \overline{1, L}, \quad (8)$$

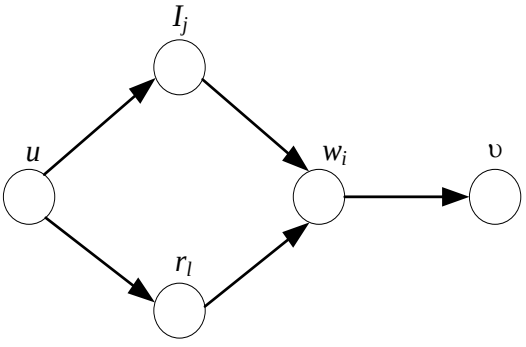
$$t_i^0 - \sum_{l=1}^{L} a_l r_{il}^2 > 0,\ \ i = \overline{1, n}. \quad (9)$$

Величины t_i^0 и r_{il}^2 определяются по аналогии с тем, как это сделано для смешанного комплекса работ.

Для решения задачи по примеру [5] построим потоковую сеть $G = (N, A)$, дополнив ее элементами, соответствующими невозобновляемым ресурсам (см. рисунок). Множество узлов определим как $N = \{u, I_j, r_l, w_i, v, j = \overline{1, p}, l = \overline{1, L}, i = \overline{1, n}\}$, где $I_j = [y_{j-1}; y_j]$, $y_0 < y_1 < \ldots < y_p$ – все различные величины b_i, f_i; $i = \overline{1, n}$; u – источник; v – сток; r_l соответствует ресурсу l-го типа. Множество дуг A определим как $A = \{(u, I_j), (u, r_l), (I_j, w_i), (r_i, w_i), (w_i, v), j = \overline{1, p}, l = \overline{1, L}, i = \overline{1, n}\}$. Дуга (l_j, w_i) вводится в сеть G в том случае, если $I_j \subseteq [b_i; f_i]$.

Пропускные способности U дуг определим следующим образом: $U(u, I_j) = m(y_{j+1} - y_j)$, $U(u, r_l) = R_l$, $U(I_j, w_i) = y_{j+1} - y_j$, $U(r_l, w_i) = a_i\ r_{il}^2$, $U(w_i, v) = t_i$, $j = \overline{1, p}$, $l = \overline{1, L}$, $i = \overline{1, n}$. В отличие от [5] будем рассматривать обобщенные сети, в некоторых внутренних узлах которых величина выходящего потока равна величине входящего потока, умноженной на некоторый коэффициент (коэффициент выигрыша), постоянный для данного узла. Так, для рассматриваемой сети G в узлах l_j и w_i выполняется условие сохранения потока, а в узле r_l коэффициент выигрыша равен a_l.

По аналогии с тем, как это сделано в [5], можно показать, что решение задачи существует только в том случае, когда максимальный поток g в сети G насыщает все дуги (w_i, v), то есть



Потоковая сеть G ($j = \overline{1, p}$, $l = \overline{1, L}$, $i = \overline{1, n}$)

Stream network G ($j = \overline{1, p}$, $l = \overline{1, L}$, $i = \overline{1, n}$)

$$(w_i, v) = t_i \quad (10)$$

при всех $i = \overline{1, n}$. Если это условие выполнено, то потоки по дугам определяют расписание выполнения работ W процессорами [5]. Потоки по дугам (r_i, w_i) определяют распределение ресурса l-го типа:

$$r_{il} = g(r_i, w_i)/a_l,\ l = \overline{1, L},\ i = \overline{1, n}. \quad (11)$$

Таким образом, для решения рассматриваемой задачи разработан следующий алгоритм.

Алгоритм 7.

Шаг 1. Построить сеть G.

Шаг 2. Найти максимальный поток g в сети G.

Шаг 3. Если выполнено условие (10), решение существует; перейти на шаг 4, в противном случае – на шаг 5.

Шаг 4. Расписание выполнения работ W определяется с помощью величин $g(I_j, w_i)$ и алгоритма упаковки [5]. Распределение невозобновляемых ресурсов определяется по формуле (11). Алгоритм завершен.

Шаг 5. Решения не существует. Алгоритм завершен.

Если при нахождении максимального потока в сети G использован алгоритм кубической сложности, то вычислительная сложность алгоритма 7 составляет $O((n + L)^3)$, поскольку число узлов в сети G равно $O(n + L)$. Отметим, что в [19] предполагается, что длительность выполнения каждой работы выражается убывающей по каждой переменной функцией от количества предоставленных этой работе невозобновляемых ресурсов. Эта задача сводится к минимизации некоторой функции с $O(n(n + L))$ переменными при $O(n^2)$ линейных ограничений. Предположение о линейности длительностей от выделенных ресурсов позволило свести задачу к потоковой.

Заключение

В статье представлены разработанные алгоритмы планирования вычислений в системах реального времени. Для работ, допускающих прерывания и переключения с одного процессора на другой, предложены более быстрые алгоритмы по сравнению с известными за счет использования подходящей структуры данных. Доказана NP-трудность задачи в случае, когда учитываются временные издержки на обработку прерываний и переключений. Для случая с отношениями предшествования предложен алгоритм, основанный на коррекции директив-

ных интервалов и использовании алгоритма относительной срочности. Для непрерываемых работ разработан псевдополиномиальный алгоритм, основанный на ограниченном переборе. Исследована задача распределения смешанного комплекса ресурсов (возобновляемых и невозобновляемых). Для случая общего директивного интервала и смешанного набора работ (прерываемых и непрерываемых) разработан псевдополиномиальный алгоритм. Для случая прерываемых работ и произвольных директивных интервалов разработан полиномиальный алгоритм, основанный на сведении исходной задачи к потоковой.

В дальнейшем авторы планируют исследовать обобщения рассмотренных задач на случаи наличия неопределенных факторов и вероятностных характеристик параметров.

Список литературы

1. 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. doi: 10.14529/cmse170404.

2. Глонина А.Б., Балашов В.В. О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов // МАИС. 2018. Т. 25. № 2. С. 174–192. doi: 10.18255/1818-1015-2018-2-174-192.

3. Глонина А.Б. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем // Вестн. МГУ. Сер. 15. Вычисл. матем. и киберн. 2020. № 3. С. 16–29.

4. Кононов Д.А., Фуругян М.Г. Региональное управление: оперативное планирование в режиме реального времени // Управление развитием крупномасштабных систем: тр. Междунар. конф. 2023. № 1. С. 1138–1144.

5. Танаев В.С., Гордон В.С., Шафранский Я.М. Теория расписаний. Одностадийные системы. М.: Наука, 1984. 383 с.

6. Brucker P. Scheduling Algorithms. Springer Publ., 2007, 383 p.

7. Кошелев П.С., Мищенко А.В. Оптимизация управления работами логистического проекта в условиях неопределенности // Изв. РАН. ТиСУ. 2021. № 4. С. 123–134. doi: 10.31857/S0002338821040089.

8. 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. Т. 6. № 6. С. 65–76. doi: 10.31857/S0002338822050079.

9. Косоруков О.А., Лемтюжникова Д.В., Мищенко А.В. Методы и модели управления ресурсами проекта в условиях неопределенности // Изв. РАН. ТиСУ. 2023. № 3. С. 38–56. doi: 10.31857/S0002338823020117.

10. Лазарев А.А. Теория расписаний. Методы и алгоритмы. М.: ИПУ РАН, 2019. 408 с.

11. Костенко В.А., Смирнов А.С. Потоковые алгоритмы планирования вычислений в интегрированной модульной авионике // Изв. РАН. ТиСУ. 2019. № 3. С. 77–86. doi: 10.1134/S0002338819030119.

12. Балашов В.В., Костенко В.А., Федоренко И.А., Гао Ц., Сун Ч.М., Сун Ц. Алгоритм имитации отжига для построения списочных расписаний с ограничениями на количество межпроцессорных передач данных // Автоматика и телемеханика. 2023. № 8. С. 138–152. doi: 10.31857/S0005231023080093.

13. Gorman M.F., Conway D.G. A tutorial of integrating duality and branch and bound in earliness–tardiness scheduling with idle insertion time problems. IJPR, 2018, vol. 56, no. 1-2, pp. 262–277. doi: 10.1080/00207543.2017.1397794.

14. Graves S.C. How to think about planned lead times. SSRN Electronic J., 2022. doi: 10.2139/ssrn.3485059. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3485059 (дата обращения: 10.07.2024).

15. Thomasson O., Battarra M., Erdoğan G., Laporte G. Scheduling twin robots in a palletising problem. IJPR, 2018, vol. 56, no. 1-2, pp. 518–542. doi: 10.1080/00207543.2017.1401249.

16. Коффман Э.Г. Теория расписаний и вычислительные машины; [пер. с англ.]. М.: Наука, 1984. 336 с.

17. Фуругян М.Г. Некоторые алгоритмы анализа и синтеза многопроцессорных вычислительных систем реального времени // Программирование. 2014. № 1. С. 36–44.

18. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ; [пер. с англ.]. М.: Вильямс, 2011. 1296 с.

19. Kononov D.A., Furugyan M.G. Distribution of heterogeneous complex of resources on a multiprocessor system. Proc. Int. Conf. SUMMA, 2023, pp. 368–372. doi: 10.1109/SUMMA60232.2023.10349570.

Software & Systems doi: 10.15827/0236-235X.149.055-064 2025, 38(1), pp. 55–64

Planning computations in real-time systems: Efficient algorithms for constructing optimal schedules

Dmitry A. Kononov [1], Meran G. Furugyan [2]✉

[1] Trapeznikov Institute of Control Sciences of the Russian Academy of Sciences, Moscow, 117997, Russian Federation

[2] Federal Research Center “Computer Science and Control” of the Russian Academy of Sciences, Moscow, 119333, Russian Federation

For citation
Kononov, D.A., Furugyan, M.G. (2025) 'Planning computations in real-time systems: Efficient algorithms for constructing optimal schedules', *Software & Systems*, 38(1), pp. 55–64 (in Russ.). doi: 10.15827/0236-235X.149.055-064
Article info
Received: 02.08.2024 After revision: 13.08.2024 Accepted: 23.08.2024

Abstract. The paper discusses issues related developing one of the main blocks of a real-time computing system, specifically the computation scheduling block. The authors propose algorithms for constructing optimal schedules for different cases depending on the number of processors and characteristics of works and computing system resources. For the single-processor case with interruptions and directive intervals, they improved the relative urgency algorithm using a heap for data storage. This contributed to lowering the algorithm computational complexity. The authors also developed an algorithm for a problem with a partial order of job execution. It bases on the pre-correction of ready moments and directive deadlines and on the reduction of the original task to a task without precedence relations. For the multiprocessor case with interruptions and directive intervals, the authors proposed an approximate algorithm that is based on a generalization of the single-processor relative urgency algorithm to the multi-processor case. The authors performed a comparative analysis with the exact stream algorithm. They proved that the problem is NP-hard when interruption and switching time costs are taken into account. For the multiprocessor case without interruptions and switches with a common directive interval for all works and identical processors, the authors developed a pseudo-polynomial algorithm based on a limited search of options. The authors also created an approximate algorithm for a system with renewable and non-renewable resources, as well as for a complex with a mixed set of works (both continuous and allowing interruptions and switching). The algorithm is based on network modeling and reducing the problem under study to the search for a stream with certain properties in a special network.
Keywords: real-time computing system, single-processor and multiprocessor systems, stream network, acceptable schedule, renewable and non-renewable resources

References

1. Glonina, A.B. (2017) 'General model of real-time modular computer systems operation for checking acceptability of such systems configurations', *Bull. of the SUrSU. Ser. Math. Mechanics. Physics*, 6(4), pp. 43–59 (in Russ.). doi: 10.14529/cmse170404.
2. Glonina, A.B., Balashov, V.V. (2018) 'On the correctness of real-time modular computer systems modeling with stopwatch automata networks', *Modeling and Analysis of Information Systems*, 25(2), pp. 174–192 (in Russ.). doi: 10.18255/1818-1015-2018-2-174-192.
3. Glonina, A.B. (2020) 'A tool system for schedulability analysis of modular computer systems configulations', *Bull. of MSU. Computational Math. and Cybernetics*, (3), pp. 16–29 (in Russ.).
4. Kononov, D.A., Furugyan, M.G. (2023) 'Situational operational planning in real-time systems', *Proc. Int. Conf. MLSD*, (1), pp. 1138–1144 (in Russ.).
5. Tanaev, V.S., Gordon, V.S., Shafranskii, Ya.M. (1984) *Theory of Scheduling: Single-Stage Systems*. Moscow, 383 p. (in Russ.).
6. Brucker, P. (2007) *Scheduling Algorithms*, Springer Publ., 378 p.
7. Koshelev, P.S., Mishchenko, A.V. (2021) 'Optimizing management of jobs in a logistic project under conditions of uncertainty', *J. of Computer and Systems Sci. Int.*, 60(4), pp. 595–609. doi: 10.1134/S1064230721040079.
8. Gorskiy, M.A., Mishchenko, A.V., Nesterovich, L.G., Khalikov, M.A. (2022) 'Some modifications of integer optimization problems with uncertainty and risk', *J. of Computer and Systems Sci. Int.*, 6(6), pp. 65–76 (in Russ.). doi: 10.31857/S0002338822050079.
9. Kosorukova, O.A., Lemtyuzhnikova, D.V., Mishchenko, A.V. (2023) 'Methods and models of project resource management under uncertainty', *J. of Computer and Systems Sci. Int.*, (3), pp. 38–56 (in Russ.). doi: 10.31857/S0002338823020117.
10. Lazarev, A.A. (2019) *Scheduling Theory. Methods and Algorithms*. Moscow, 408 p. (in Russ.).
11. Kostenko, V.A., Smirnov, A.S. (2019) 'Flow algorithms for scheduling computations in integrated modular avionics', *J. of Computer and Systems Sci. Int.*, 58(3), pp. 404–414. doi: 10.1134/S1064230719030110.
12. Balashov, V.V., Kostenko, V.A., Fedorenko, I.A., Gao, C., Sun, C.M., Sun, C. (2023) 'Simulated annealing algorithm for constructing list schedules with restrictions on the number of interprocessor data transfers', *Automation and Remote Control*, (8), pp. 138–152 (in Russ.). doi: 10.31857/S0005231023080093.
13. Gorman, M.F., Conway, D.G. (2018) 'A ttutorial of integrating duality and branch and bound in earliness–tardiness scheduling with idle insertion time problems', *IJPR*, 56(1-2), pp. 262–277. doi: 10.1080/00207543.2017.1397794.
14. Graves, S.C. (2022) 'How to think about planned lead times', *SSRN Electronic J*. doi: 10.2139/ssrn.3485059, available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3485059 (accessed July 10, 2024).
15. Thomasson, O., Battarra, M., Erdoğan, G., Laporte, G. (2018) 'Scheduling twin robots in a palletising problem', *IJPR*, 56(1-2), pp. 518–542. doi: 10.1080/00207543.2017.1401249.
16. Coffman, E.G. (1976) *Computer and Job-Shop Scheduling Theory*. John Wiley & Sons Publ., 299 p. (Russ. ed.: (1984) Moscow, 336 p.).

17. Furugyan, M.G. (2014) 'Some algorithms for analysis and synthesis of real-time multiprocessor computing systems', *Programming and Computer Software*, 40(1), pp. 21–27. doi: 10.1134/S0361768814010034.

18. Cormen, T.H., Leiserson, Ch.E., Rivest R.L., Stein, C. (2011) *Introduction to Algorithms*. The MIT Press, 1312 p. (Russ. ed.: (2011) Moscow, 1296 p.).

19. Kononov, D.A., Furugyan, M.G. (2023) 'Distribution of heterogeneous complex of resources on a multiprocessor system', *Proc. Int. Conf. SUMMA*, pp. 368–372. doi: 10.1109/SUMMA60232.2023.10349570.

Авторы

Кононов Дмитрий Алексеевич [1], д.т.н., доцент, ведущий научный сотрудник, dmitrykon52@gmail.com

Фуругян Меран Габибуллаевич [2], к.ф.-м.н., доцент, старший научный сотрудник, rtsccas@yandex.ru

[1] Институт проблем управления им. В.А. Трапезникова РАН, г. Москва, 117997, Россия

[2] Федеральный исследовательский центр «Информатика и управление» РАН, г. Москва, 119333, Россия

Authors

Dmitry A. Kononov [1], Dr.Sci. (Engineering), Associate Professor, Leading Researcher, dmitrykon52@gmail.com

Meran G. Furugyan [2], Cand. of Sci. (Physics), Associate Professor, Senior Researcher, rtsccas@yandex.ru

[1] Trapeznikov Institute of Control Sciences of the Russian Academy of Sciences, Moscow, 117997, Russian Federation

[2] Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, 119333, Russian Federation