Обобщение задачи составления многопроцессорного расписания с прерываниями

М.Г. Фуругян[1]

[1]ФИЦ ИУ РАН, Москва, Россия, rtsccas@yandex.ru

Аннотация. Рассматривается задача составления многопроцессорного расписания для комплекса работ, допускающих прерывания и переключения с одного процессора на другой. Предполагается, что обработка прерываний и переключений требует временных издержек. Это условие переводит задачу из класса полиномиально разрешимых в класс NP-трудных. Разработан алгоритм, основанный на методике В.С. Танаева составления многопроцессорного расписания без учета затрат на прерывания и переключения. Методика включает в себя процедуру упаковки для случая, когда работы имеют общий директивный срок, а также процедуру сведения исходной задачи к потоковой для случая произвольных директивных интервалов. При этом используется также известный псевдополиномиальный алгоритм составления допустимого многопроцессорного расписания для непрерываемых работ с общим директивным сроком.

Ключевые слова: многопроцессорная система, директивный интервал, допустимое расписание, процедура упаковки, потоковая сеть

1. Введение

По вопросам планирования работ и составления расписаний имеется большое число публикаций. Отметим такие фундаментальные работы, как [1, 2]. В [1] рассматриваются различные задачи по теории расписаний как для однопроцессорных, так и для многопроцессорных систем. Авторы предлагают алгоритмы планирования непрерываемых работ, а также для работ, допускающих прерывания и переключения с одного процессора на другой. Большое внимание уделено вопросам вычислительной сложности алгоритмов. Проводится большой список NP-трудных задач. В [2], помимо задач составления расписаний, исследуются различные задачи дискретной оптимизации.

В [3] исследуются задачи составления однопроцессорных расписаний для непрерываемых работ с критерием минимизации максимального запаздывания. Автором введено понятие расстояния между задачами, что позволило разработать эффективные алгоритмы.

В [4, 5] на основе метода ветвей и границ разработаны алгоритмы решения задач планирования в финансовой и экономической сферах. Эти публикации представляют особый интерес, поскольку в них рассмотрены задачи с нефиксированными параметрами, такими, как длительности выполнения работ и объемы имеющихся ресурсов.

В [6, 7] рассмотрены задачи планирования работ в многопроцессорных и многоядерных системах реального времени. В таких системах существуют жесткие временные ограничения на выполнения работ, которые в ряде случаев составляют доли секунды. Поэтому очень важным является составление оптимальных по быстродействию расписаний. Для этого авторы используют аппарат сетей Петри с остановкой таймера и временные диаграммы.

В работах [1 – 7] используемые ресурсы являются не складируемыми, т.е. такими, которые могут использоваться многократно. К таким ресурсам относятся, например, машины, станки, приборы. В отличие от них, складируемые ресурсы повторно использоваться не могут. К таким ресурсам относятся, например, финансы, горючие материалы, электроэнергия.

В [8, 9] исследованы задачи планирования работ в системах с неоднородным комплексом ресурсов, который включает в себя как складируемые, так и не складируемые ресурсы. Методика решения таких задач основана на их сведении к потоковым задачам в сетях специального вида.

В настоящей статье рассматривается обобщение задачи составления многопроцессорного расписания с директивными интервалами, допускающего прерывания и переключения. В отличие от указанных выше публикаций, предполагается, что прерывания и переключения требуют временных затрат. Наличие этого условия приводит к тому, что задача переходит из класса полиномиально разрешимых задач в класс NP-трудных задач. Решение задачи основано на методике, предложенной в [1] и состоящей из построения сети специального вида и поиска в ней

максимального потока. Дополнительно используется псевдополиномиальный алгоритм составления расписания выполнения непрерываемых работ с общим директивным сроком [10].

2. Постановка задачи

Для выполнения комплекса работ (заданий) $W = \{w_1, w_2, \dots, w_n\}$ имеется m идентичных процессоров $p_1, p_2, \dots, p_m$. Каждая работа может выполняться любым процессором. Для работы w_i известна длительность ее выполнения t_i и директивный интервал $[b_i, f_i]$ (работа w_i может быть начата не ранее момента времени b_i и должна быть завершена не позднее момента времени f_i), $i = \overline{1,n}$.При выполнении работ допускаются их прерывания и переключения с одного процессора на другой. В отличие от [1], предполагается, что прерывание и переключение с одного процессора на другой требуют дополнительных временных затрат в объеме σ каждого из этих двух процессоров. А именно, если некоторая работа выполняется процессором p_{l_1} и в момент времени τ_1 она прерывается, то этот процессор выполняет дополнительную работу в интервале $[\tau_1;\ \tau_1 + \sigma]$. Далее, прерванная работа может быть возобновлена в момент времени $\tau_1 \geq \tau_1 + \sigma$ на некотором процессоре p_{l_2} (возможно, на том же процессоре p_{l_1}), который сначала выполняет дополнительную работу в интервале $[\tau_2;\ \tau_2 + \sigma]$, а затем прерванную работу. Предполагается, что

$$t_i + 2\sigma \leq f_i - b_i \qquad (1)$$

при всех $i = \overline{1,n}$, т. е. каждая работа может быть выполнена одним процессором в своем директивном интервале, включая обработку одного прерывания и одного возобновления прерванной работы.

Не допускается параллельное выполнение одной работы несколькими процессорами и параллельное выполнение нескольких работ одним процессором.

Расписание выполнения комплекса работ *W* показывает для каждой работы, в какие моменты времени какими процессорами она выполняется. Допустимое расписание – это такое расписание, при котором каждая работа полностью выполняется в своем директивном интервале.

Задача состоит в том, чтобы определить, существует ли допустимое расписание выполнения комплекса работ *W*, и построить его в случае положительного ответа. Отметим, что в [11] был разработан алгоритм для случая, когда директивные интервалы всех работ совпадают. Там же показано, что сформулированная задача является NP-трудной даже в том случае, когда директивные интервалы у всех работ совпадают.

3. Модификация алгоритма упаковки

Рассмотрим сначала случай, когда директивные интервалы у всех работ совпадают, т.е. $b_i = 0, f_i = F$ при всех $i = \overline{1,n}$. В [1] описан алгоритм упаковки для случая, когда затраты на прерывания и переключения не учитываются. Ниже приводится модификация этого алгоритма для случая, когда прерывания и переключения требуют временных издержек, как это описано в разд. 2. В этом случае условие (1) трансформируется в неравенство

$$t_i + 2\sigma \leq F \qquad (2)$$

при всех $i = \overline{1,n}$.

Лемма. 1) Необходимым условием существования допустимого расписания является выполнение неравенства

$$\sum_{i=1}^{n} t_i \leq mF. \qquad (3)$$

2) Достаточным условием существования допустимого расписания является выполнение неравенства

$$\sum_{i=1}^{n} t_i + 2(m-1)\sigma \leq mF. \qquad (4)$$

Доказательство. 1) Если неравенство (3) не выполнено, то это означает, что суммарная длительность работ *W* превосходит суммарное процессорное время. В этом случае допустимого расписания не существует.

2) Существование допустимого расписания при выполнении неравенства (4) следует из описанного ниже модифицированного алгоритма упаковки. Лемма доказана.

Модифицированный алгоритм упаковки

Шаг 1. Выполнить работу w_1 на процессоре p_1 без прерываний в интервале $[o;\ t_1]$. Положить $\tau = t_2, i = 1, j = 1$.

Шаг 2. Положить $i = i + 1$. Если $i \leq n$, то перейти на шаг 3; в противном случае перейти на шаг 8.

Шаг 3. Если $\tau + t_i \leq F$, то перейти на шаг 4. Если $\tau + t_i = F$, то перейти на шаг 5. Если $\tau + t_i > F$ и $\tau + \sigma \geq F$, то перейти на шаг 7.

Шаг 4 Выполнять работу w_i на процессоре p_j в интервале $[\tau;\ \tau + t_i]$. Перейти на шаг 2.

Шаг 5. Выполнять работу w_i на процессоре p_j в интервале $[\tau;\ \tau + t_i]$. Положить $j = j + 1$. Перейти на шаг 2.

Шаг 6. Выполнять работу w_i на процессоре p_{j+1} в интервале $[0;\ t_i - (F - \tau + \sigma)]$. В момент

$t_i - (F - \tau + \sigma)$ выполнение работы w_i прерывается. Далее, в интервале $[t_i - (F - \tau - \sigma);\ t_i - (F - \tau - \sigma) + \sigma]$ процессор p_{j+1} выполняет дополнительную работу по обработке прерывания и переключения работы w_i. Далее, в интервале $[\tau;\ \tau + \sigma]$ процессор p_j снова выполняет дополнительную работу по обработке прерывания и переключения работы w_i. После чего процессор p_j возобновляет выполнение работы w_i в интервале $[\tau + \sigma; F]$. Положить $j = j + 1$. Перейти на шаг 2.

Шаг 7. Выполнять работу w_i без прерываний на процессоре p_{j+1} в интервале $[0;\ t_i]$. $j = j + 1$. Перейти на шаг 2.

Шаг 8. Расписание построено. Завершение алгоритма.

Дадим некоторые пояснения к описанному алгоритму. Выполнение работы w_1 на процессоре p_1 без прерываний и переключений (шаг 1) возможно в силу условия (2). На шаге 2 выполняется проверка, все ли работы назначены на процессоры. На шаге 3 выполняется проверка возможности исполнения очередной работы на текущем процессоре. На шаге 4 и шаге 5 очередная работа полностью выполняется на текущем процессоре, если она может завершиться не позднее момента времени *F*. В противном случае возможны два варианта. В первом варианте она сначала выполняется на следующем процессоре, а завершается на текущем процессоре (шаг 6). При этом учитываются затраты на прерывания и переключения. Во втором варианте текущая работа полностью выполняется на следующем процессоре без прерываний и переключений (шаг 7). Шаг 8 завершает алгоритм.

Отметим, что в результате работы писанного алгоритма число прерываний и переключений не более $m - 1$, а продолжительность их обработки не превосходит $2(m - 1)\sigma$. Поэтому если условие (2) будет выполнено, то это гарантирует существование допустимого расписания. Вычислительная сложность описанного алгоритма составляет $O(n)$.

4. Сокращение числа прерываний и переключений для случая одинаковых директивных интервалов

В предыдущем разделе предполагалось, что выполняется неравенство (4), которое гарантирует существование допустимого расписания в случае общего директивного срока у всех работ. В настоящем разделе будем предполагать, что неравенство (4) не выполняется, т.е.

$$\sum_{i=1}^{n} t_i + 2(m-1)\sigma > mF \qquad (5)$$

а неравенство (3) выполняется, т.к. оно является необходимым условием существования допустимого расписания.

Упорядочим работы по не убыванию их длительностей, т.е. будем предполагать, что $t_1 \le t_2 \le \cdots \le t_n$, Предлагаемый алгоритм заключается в том, чтобы сократить число прерываний и переключений. Для этого множество процессоров разбивается на два подмножества: $P_1 = \{p_1, \dots, p_k\}$ и $P_2 = \{p_{K+1}, \dots, p_m\}$. Используя алгоритм, описанный в [10]. Построим допустимое расписание без прерываний и переключений на процессорах P_1 для некоторого подмножества работ $W_1 \subseteq W$. Это расписание получается путем последовательного выбора работ из W и построения точек в *k*-мерном кубе с ребром длины *F*. Вычислительная сложность этого алгоритма $O(kT^k)$. Если для работ $W_2 = W \backslash W_1$ и $m - k$ процессоров выполняется условие существования допустимого расписания, т.е. неравенство

$$\sum_{i \epsilon W_2} \boldsymbol{t_i} + \boldsymbol{2(m-k-1)\sigma \le (m-k)F}, \qquad \boldsymbol{(6)}$$

то с помощью модифицированного алгоритма упаковки построим допустимое расписание для W_2. Объединяя его с ранее построенным расписанием для W_1, построим окончательное допустимое расписание для W. Если же неравенство (6) не выполняется, то число *k* следует увеличить на одну единицу. Таким образом, алгоритм выглядит следующим образом.

Шаг 1. Положить $k = 1$.

Шаг 2. Построить допустимое расписание без прерываний и переключений на процессорах P_1. Пусть W_1 – множество работ, вошедшее в это расписание.

Шаг 3. Если выполнено неравенство (6), то построить для W_2 расписание на процессорах P_2 с прерываниями и переключениями; завершение алгоритма. Если неравенство (6) не выполнено, то перейти на шаг 4.

Шаг 4. Если $k = m$, то решение не найдено; завершение алгоритма. Если $k < m$, то положить $k = k + 1$ и перейти на шаг 2.

Вычислительная сложность описанного алгоритма $O(m(T^m + n))$.

5. Произвольные директивные интервалы

Перейдем к рассмотрению случая произвольных директивных интервалов. Сначала приведем краткое описание алгоритма, предложенного в [1] в предположении отсутствия издержек

на обработку прерываний и переключений.

Пусть $y_0 < y_1 < \cdots < y_p$ – все различные значения $b_i, f_i, i = \overline{1,n}$; $I_j = [y_{j-1};\ y_j], \delta_j = y_j - y_{j-1},\ j = \overline{1,p}$. Определим потоковую сеть $G = (V, A)$, V – множество вершин, A – множество ориентированных дуг; $V = \{u, v, I_j, w_i\}$, u – источник, v – сток, $A = \{(u, I_j), (I_j, w_i), (w_i, v)\}, j = \overline{1,p},\ i = \overline{1,n}$. Дуга (I_j, w_i) включается в A, если $I_j \subseteq [b_i;\ f_i]$. Пропускные способности U дуг определяются следующим образом: $U(u, I_j) = m\delta_j$, $U(I_j, w_i) = \delta_j$, $U(w_i, v) = t_i$. В [1] доказано, что допустимое расписание существует в том и только том случае, когда в сети G существует поток g, для которого $g(w_i, v) = t_i$ при всех $i = \overline{1,n}$. Если такой поток существует и $g(I_j, w_i) > 0$, то работе w_i в интервале I_j выделяется $g(I_j, w_i)$ единиц процессорного времени. Расписание для каждого интервала I_j строится с помощью модифицированного алгоритма упаковки, описанного в разделах 3, 4.

6. Сокращение числа прерываний и переключений для случая произвольных директивных интервалов

Как следует из разд. 4, прерывания и переключения, помимо возникающих внутри каждого интервала $I_j, j = \overline{1,p}$, возможны и на стыках интервалов I_j и $I_{j+1}, j = \overline{1,p} - 1$. В этом разделе будем исследовать вопрос о сокращении числа таких прерываний и переключений.

Как следует из разд. 2, 3, расписание внутри каждого интервала $I_j, j = \overline{1,p}$, может быть представлено в виде матрицы $\|w_{ki_r}^j\|$, $k = \overline{1,m}$, $1 \le r \le n$. Здесь $w_{ki_r}^j$ – работа w_{i_r}, выполняемая в интервале I_j процессором p_k. Кроме того, при $k = \overline{1,k(j)}$, $k(j) \le m$, работы выполняются без прерываний и переключений.

Предлагаемый алгоритм состоит в следующем. Рассмотрим пару интервалов I_j и $I_{j+1}, j = \overline{1,p} - 1$. Предположим, что $w_{1i_1}^j = w_{k_2i_2}^{j+1}$ при некоторых $i_1, i_2, 1 \le k_2 \le k(j+1)$. Тогда меняем местами строки 1 и k_2 в матрице $\|w_{ki_r}^{j+1}\|$. Далее, работу $w_{1i_1}^j$ выполняем в конце интервала I_j, а работу $w_{k_2i_2}^{j+1}$ – в начале интервала I_{j+1}. В этом случае работа $w_{1i_1}^j$ будет выполняться без прерываний и переключений в пределах интервалов I_j и I_{j+1}.

Далее, следует рассмотреть элементы $w_{1i_2}^j$ и $w_{k_ri_r}^{j+1}$ при $2 \le k_2 \le k(j+1)$ и выполнять действия аналогичным образом.

Вычислительная сложность описанной процедуры составляет $O(n^2p)$ или $O(n^3)$.

7. Заключение

Исследована задача составления многопроцессорного расписания для комплекса работ, допускающих прерывания и переключения с одного процессора на другой. Предполагается, что обработка прерываний и переключений требует временных издержек, в следствие чего задача является NP-трудной. Разработанный алгоритм основан на методике В.С. Танаева составления многопроцессорного расписания без учета затрат на прерывания и переключения. Алгоритм включает в себя: процедуру упаковки для случая, когда работы имеют общий директивный срок, процедуру сведения исходной задачи к потоковой для случая произвольных директивных интервалов, псевдополиномиальный алгоритм составления допустимого многопроцессорного расписания для непрерываемых работ с общим директивным сроком. Для отдельных частей алгоритма получены оценки вычислительной сложности.

Generalization of the Problem of Creating a Multiprocessor Schedule with Interrupts

Meran Furugyan

Abstract. We consider the problem of creating a multiprocessor schedule for a set of jobs that allow interruptions and switching from one processor to another. It is assumed that processing interrupts and switches requires time overhead. This condition transfers the problem from the class of polynomially solvable to the class of NP-hard ones. An algorithm has been developed based on the methodology of V.S. Tanaev for compiling a multiprocessor schedule without taking into account the costs of interruptions and switching. The technique includes a packing procedure for the case when jobs have a common deadline, as well as a procedure for reducing the original problem to a network flow problem for the case of arbitrary deadlines. In this case, the well-known pseudo-polynomial algorithm for creating an admissible multiprocessor schedule for continuous jobs with a common deadline is also used.

Keywords: multiprocessor system, admissible schedule, directive interval, packing procedure, flow network

Литература

1. Танаев В.С., Гордон В.С., Шафранский Я.М. Теория расписаний. Одностадийныесистемы. М.: Наука, 1984, 383 с.

2. Brucker P. Scheduling Algorithms. Heidelberg: Springer, 2007, 378 с.

3. Лазарев А.А. Теория расписаний. Методы и алгоритмы. –М.: ИПУ РАН, 2019,407 с.

4. А.В. Мищенко, П.С. Кошелев. Оптимизация управления работами логистического проекта в условиях неопределенности // Известия РАН. Теория и системы управления. (2021), № 4, 123-134.

5. М.А. Горский, А.В. Мищенко, Л.Г. Нестерович, М.А. Халиков. Некоторые модификации целочисленных оптимизационных задач с учетом неопределенности и риска // Известия РАН. Теория и системы управления. (2022), № 5, 106-117.

6.А.Б. Глонина, В.В. Балашов. О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов // Моделирование и анализ информационных систем. (2018), Т. 25, № 2, 174 – 192.

7. А.Б. Глонина. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем // Вестн. МГУ. Сер. 15. Вычисл. математика и кибернетика. (2020), № 3, 16 – 29.

8. М.Г. Фуругян. Планирование вычислений в многопроцессорных АСУ реального времени с дополнительным ресурсом //Автоматика и телемеханика. (2015), №3, 144 – 150.

9. М.Г. Фуругян. Составление расписаний в многопроцессорных системах с несколькими дополнительными ресурсами // Известия РАН. Теория и системы управления. (2017), № 2, 57 – 66.

10. М.Г. Фуругян. Некоторые алгоритмы решения минимаксной задачи составления многопроцессорного расписания // Известия РАН. Теория и системы управления. (2014), №2, 50 - 56.

11. М.Г. Фуругян. Составление расписаний в многопроцессорной системе с учетом затрат на прерывания // В Сб. Математическое и компьютерное моделирование сложных систем: теоретические и прикладные аспекты. Труды НИИСИ РАН. Т. 6, № 2, 57 – 61.