УДК 65.012.122

СОСТАВЛЕНИЕ РАСПИСАНИЙ РЕШЕНИЯ ЗАДАЧ В КОНВЕЙЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Н. В. Колесов,
доктор техн. наук, профессор
М. В. Толмачева,
ведущий инженер
ГНЦ РФ ЦНИИ «Электроприбор» (Санкт-Петербург)

Рассматриваются алгоритмы составления расписаний в конвейерных вычислительных системах. Анализируются три базовых случая, для которых указываются бесперебо́рные алгоритмы составления расписаний для случаев, когда времена решения задач точно известны и когда они задаются интервалами.

The algorithms of Schedule tabling for tasks solution in pipeline computer systems are considered. Three basic cases are analyzed. Author proposes algorithms, which are used for the cases with certain time of task solution specified as intervals.

Введение

Проблема составления расписаний возникает во многих приложениях и, в частности, при формировании последовательности исполняемых задач в многопроцессорных вычислительных системах. Эти системы могут характеризоваться различной структурой – последовательной, параллельной, последовательно-параллельной, параллельно-последовательной. В каждом случае задача имеет свои особенности. Ниже рассматриваются системы с последовательной структурой, или конвейерные системы. Наиболее часто проблема формулируется как поиск такой упорядоченности (расписания) для заданного списка задач, при которой время решения всех задач из списка минимально. Известно [1–3], что решение таких задач в общем случае характеризуется высокой алгоритмической сложностью, описываемой экспоненциальной функцией от длины списка. Такие задачи называют задачами неполиномиальной (экспоненциальной) сложности, или коротко NP-сложными задачами. Однако в некоторых так называемых разрешимых случаях можно предложить более простые алгоритмы. Ряд таких случаев описан, например, в книге [2]. В настоящей статье также анализируются некоторые из таких разрешимых случаев, рассматриваемые при произвольном числе процессоров и задач. Задачи рассматриваются как при условии, что времена решения задач известны точно, так и при условии, что они известны приблизительно и задаются временными интервалами. При этом показывается возможность приближенного сведения задач из расширенного класса к рассматриваемым базовым разрешимым случаям.

Постановка задачи. Базовые разрешимые случаи

Далее будем предполагать, что рассматриваемая вычислительная система представляет собой конвейер из m процессоров $L = \{L_1, L_2, \ldots, L_m\}$, на вход которого поступает последовательность $(J_1, J_2, \ldots, J_n)$ из n задач. Каждая задача разбивается на m фрагментов (по числу процессоров). При этом i-й фрагмент j-й задачи решается на i-м процессоре за время $\tau_{i,j}$ $i = 1, m$, $j = 1, n$. Проблема состоит в определении такой последовательности (расписания) решения задач, которая требует минимального суммарного времени.

Введем на множестве процессоров отношение доминирования «$>$».

Определение 1. Процессор L_q доминирует над процессором L_r ($L_q > L_r$), если $\min\limits_j \tau_{q,j} \geq \max\limits_j \tau_{r,j}$.

Рассмотрим три базовых случая составления расписаний. Во всех случаях на множестве процессоров можно выделить возрастающую или убывающую по отношению доминирования последовательность.

Случай 1. Множество процессоров представляет собой последовательность, убывающую по отношению доминирования, а именно:

$$L_1 > L_2 > \ldots > L_m. \quad (1)$$

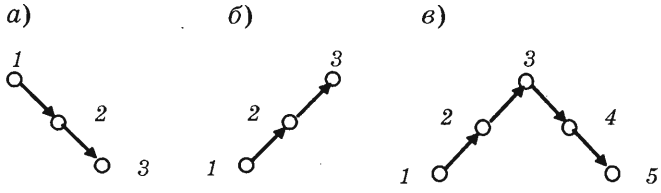
Случай 2. Множество процессоров представляет собой последовательность, возрастающую по отношению доминирования, а именно:

$$L_1 < L_2 < \ldots < L_m. \quad (2)$$

Случай 3. Множество процессоров состоит из пары соединенных последовательностей, первая из которых возрастает, а вторая убывает по отношению доминирования, а именно:

$$L_1 < L_2 < \ldots < L_h > \ldots > L_{m-1} > L_m,\ 1 \le h \le m. \quad (3)$$

На рис. 1 для наглядности приведена графическая интерпретация рассматриваемых случаев. При этом вершины графа отображают процессоры, ребра указывают направленность конвейера, а отношение доминирования между процессорами отражается взаимным расположением соответствующих вершин (доминирующая вершина располагается выше).



■ ***Рис. 1.*** *Графическая интерпретация базовых случаев (а – случай 1, б – случай 2, в – случай 3)*

Решение детерминированной задачи для базовых случаев

Покажем теперь, как сконструировать расписание в первом случае.

Алгоритм 1.

1. Выделим задачу J_s, которая удовлетворяет условию

$$\sum_{i=2}^{m} \tau_{i,s} = \min_j \{\sum_{i=2}^{m} \tau_{i,j}\}.$$

2. Сформируем расписание $\pi_1 = [\tilde{\pi}\, J_s]$, где $\tilde{\pi}$ – произвольное расписание для $(n-1)$ задач, не содержащее задачи J_s.

Покажем оптимальность алгоритма 1. Предварительно отметим важную особенность рассматриваемого случая, заключающуюся в том, что ни перед одним из процессоров конвейера никогда не образуется очередь из задач, ожидающих решения. Этот факт следует из определения отношения доминирования, в соответствии с которым в первом случае самая короткая задача будет решена на некотором процессоре за большее время, чем самая длинная задача на любом последующем процессоре.

Запишем выражение для времени реализации произвольного расписания при выполнении условия (1). Учитывая, что перед процессорами не образуется очередей и все задачи переходят с процессора на процессор без задержек, время $T_1(\pi)$ реализации произвольного расписания π можно представить в виде суммы времени ожидания в исходной очереди для последней n-й задачи и времени t_n ее решения. Причем время ожидания будет определяться суммой времен решения первых фрагментов всех задач, кроме последней. В результате имеем:

$$T_1(\pi) = \sum_{j=1}^{n-1} \tau_{1,j} + t_n = \sum_{j=1}^{n-1} \tau_{1,j} + \sum_{i=1}^{m} \tau_{i,n}.$$

Перенося для удобства анализа первое слагаемое из второй суммы в первую, получаем:

$$T_1(\pi) = \sum_{j=1}^{n} \tau_{1,j} + \sum_{i=2}^{m} \tau_{i,n}. \quad (4)$$

Значение первой суммы представляет собой суммарное время, затрачиваемое на решение первых фрагментов всех задач. Очевидно, что для заданной очереди ее величина фиксирована и не зависит от выбора расписания. В отличие от первой суммы вторая зависит от выбора расписания. Причем оптимальным будет то расписание, которое характеризуется минимальным значением этой суммы. Отсюда следует, что расписание, формируемое в алгоритме 1, является оптимальным для случая 1.

Отметим два существенных обстоятельства. Во-первых, в данном алгоритме отсутствует перебор вариантов расписаний. Во-вторых, для оптимальности расписания достаточно лишь правильного выбора последней исполняемой задачи, которая должна быть наиболее быстро решаемой на всех процессорах, возможно, кроме первого. Упорядоченность же остальных задач не влияет на время исполнения расписания.

Рассмотрим правила формирования расписания во втором случае, который в определенном смысле противоположен первому. Он также характеризуется определяющим свойством, но оно заключается в том, что перед каждым из процессоров всегда образуется очередь из фрагментов задач, ожидающих решения. В результате каждая задача, кроме первой, будет стартовать на любом процессоре с некоторой задержкой относительно момента ее прибытия на вход процессора, поскольку любой i-й фрагмент j-й задачи будет заканчиваться позже, нежели $(i-1)$-й фрагмент следующей за ней $(j+1)$-й задачи. Этот факт также следует из определения отношения доминирования. Алгоритм конструирования расписания в этом случае будет выглядеть следующим образом.

Алгоритм 2.

1. Выделим задачу J_l, которая удовлетворяет условию

$$\sum_{i=1}^{m-1} \tau_{il} = \min_j \{\sum_{i=1}^{m-1} \tau_{i,j}\}.$$

2. Сформируем расписание $\pi_2 = [J_l \tilde{\pi}]$, где $\tilde{\pi}$ – произвольное расписание для $(n-1)$ задач, не содержащее задачи J_l.

Покажем оптимальность этого алгоритма. Запишем выражение для времени реализации произвольного расписания при выполнении условия (2). Учтем, что перед последним n-м процессором (как и перед всеми другими) всегда существует очередь, а значит, этот процессор всегда занят с момента появления на его входе первой и до момента ухода с него последней задачи. В связи с этим время $T_2(\pi)$ реализации произвольного расписания π можно представить в виде суммы времени t_1, необходимого для решения первой задачи, и времени решения последних m-х фрагментов всех остальных задач:

$$T_2(\pi) = t_1 + \sum_{j=2}^{n} \tau_{m,j} = \sum_{i=1}^{m} \tau_{i,1} + \sum_{j=2}^{n} \tau_{m,j}.$$

Перенося для удобства анализа последнее слагаемое из первой суммы во вторую, получаем:

$$T_2(\pi) = \sum_{i=1}^{m-1} \tau_{i,1} + \sum_{j=1}^{n} \tau_{m,j}. \qquad (5)$$

Значение второй суммы представляет собой суммарное время, необходимое для решения последних фрагментов всех задач. Очевидно, что для заданной очереди ее величина фиксирована и не зависит от выбора расписания. В отличие от второй суммы первая зависит от выбора расписания. Причем оптимальным будет то расписание, которое характеризуется минимальным значением этой суммы. Отсюда следует, что расписание, формируемое в алгоритме 2, является оптимальным для случая 2.

Так же, как и в предыдущем случае, в данном алгоритме отсутствует перебор вариантов расписаний, а для оптимальности расписания достаточно лишь правильного выбора первой исполняемой задачи, которая должна быть наиболее быстро решаемой на всех процессорах, возможно, кроме последнего. Упорядоченность же остальных задач не влияет на время исполнения расписания.

Рассмотрим задачу построения оптимального расписания в случае 3.

Алгоритм 3.

1. Выделим задачу J_s, которая удовлетворяет условию

$$\sum_{i=1}^{h-1} \tau_{i,s} = \min_j \{\sum_{i=1}^{h-1} \tau_{i,j}\}.$$

2. Выделим задачу J_p, которая удовлетворяет условию

$$\sum_{i=h+1}^{m} \tau_{i,p} = \min_{\substack{j \\ j \neq s}} \{\sum_{i=h+1}^{m} \tau_{i,j}\}.$$

3. Сформируем расписание $\pi_3 = [J_s \tilde{\pi} J_p]$, где $\tilde{\pi}$ – произвольное расписание для $(n-2)$ задач, не содержащее задач J_s и J_p.

4. Сформируем расписание $\pi_4 = [J_q \tilde{\pi} J_r]$, повторив пп. 1–3, но выполнив пп. 1 и 2 в другой последовательности.

5. Выберем из расписаний π_3 и π_4 наилучшее.

Покажем оптимальность этого алгоритма. Разобьем конвейер на две части. В первую часть включим процессоры с первого по $(h-1)$-й, а во вторую – с h-го по m-й. Очевидно, что первая часть конвейера соответствует случаю 2. В результате каждая задача перед решением на любом процессоре в этой части конвейера попадает в очередь. Поскольку то же самое справедливо и для h-го процессора, можно утверждать, что вторая часть конвейера соответствует случаю 1. Действительно, не только выполняется условие (1), но и задачи выходят на эту часть конвейера без каких-либо дополнительных задержек так, как будто они все одновременно стоят в очереди к h-му процессору. Запишем время $T_3(\pi)$ исполнения произвольного расписания на рассматриваемом конвейере. Представим искомое время $T_3(\pi)$ как сумму двух величин. Первая величина – это время $t_n(1, h-1)$ от начала решения первой задачи на первом процессоре до начала решения n-й задачи на h-м процессоре, вторая величина – это время $t_n(h, m)$ решения n-й задачи на второй части конвейера:

$$T_3(\pi) = t_n(1, h-1) + t_n(h, m).$$

Очевидно, что время $t_n(1, h-1)$ будет равняться времени решения $(n-1)$-й задачи на процессорах с 1-го по h-й, которое определяется выражением (5) и имеет вид

$$t_n(1, h-1) = t_{n-1}(1, h) = \sum_{i=1}^{h} \tau_{i,1} + \sum_{j=2}^{n-1} \tau_{h,j}.$$

Время $t_n(h,m)$ решения n-й задачи на второй части конвейера определяется очевидным выражением

$$t_n(h, m) = \sum_{i=h}^{m} \tau_{i,n}.$$

В результате для $T_3(\pi)$ получаем:

$$T_3(\pi) = \sum_{i=1}^{h} \tau_{i,1} + \sum_{j=2}^{n-1} \tau_{h,j} + \sum_{i=h}^{m} \tau_{i,n}.$$

Перенося для удобства анализа последнее слагаемое из первой суммы и первое слагаемое из третьей суммы во вторую, получаем:

$$T_3(\pi) = \sum_{i=1}^{h-1} \tau_{i,1} + \sum_{j=1}^{n} \tau_{h,j} + \sum_{i=h+1}^{m} \tau_{i,n}. \qquad (6)$$

Значение второй суммы представляет собой суммарное время, затрачиваемое на решение всех задач на h-м процессоре. Очевидно, что для заданной очереди ее величина фиксирована и не зависит от выбора расписания. Первая и третья суммы, в отличие от второй, зависят от выбора расписания. Причем оптимальным будет то расписание, которое характеризуется минимальным значением этой части выражения. Минимальность значения, в свою очередь, будет достигаться, если в расписании первой будет решаться задача, требующая наименьшего времени для первых $(h-1)$ фрагментов, а последней – задача, наиболее быстро решаемая на последних $(m-h)$ процессорах. Поскольку одна и та же задача может удовлетворять обоим требованиям, надо попробовать разместить ее на каждой из этих позиций, а затем выбрать наилучшее из этих двух расписаний. Отсюда следует, что расписание, формируемое в алгоритме 3, является оптимальным для случая 3.

Отметим, что в данном алгоритме перебор расписаний также практически отсутствует. При построении расписания достаточно проанализировать лишь два возможных варианта, а для оптимальности расписания достаточно лишь правильного выбора первой и последней исполняемых задач. Упорядоченность же остальных задач не влияет на время исполнения расписания.

Пример. Построим расписание для конвейерной системы, для которой времена решения задач, выраженные в условных единицах, приведены в табл. 1.

Нетрудно заметить, что данная система удовлетворяет условию (3). Тогда в соответствии с алгоритмом 3 для составления расписания для каждой задачи следует вычислить две суммы:

$$S_1 = \sum_{i=1}^{h-1} \tau_{i,1}; \quad S_2 = \sum_{i=h+1}^{m} \tau_{i,n}.$$

Результаты вычислений приведены в табл. 2.

Из таблицы видно, что первая сумма принимает минимальное значение в первой задаче J_1, а вторая – в четвертой J_4. Следовательно, эти задачи должны занимать соответственно первую и последнюю позиции в оптимальном расписании. Таким образом, в данном случае имеются два варианта оптимального расписания: $\pi_1 = J_1J_2J_3J_4$ и $\pi_2 = J_1J_3J_2J_4$. На рис. 2 приведен график Ганта, соответствующий построенному оптимальному расписанию π_1. На этом графике по оси ординат отложены номера задач, а по оси абсцисс – время, представленное в условных единицах. Для каждой из задач временные интервалы исполнения ее фрагментов на разных процессорах выделены чередующимися цветами. Как видно, время исполнения всего расписания равно 45, что совпадает с расчетным результатом, полученным из выражения (6).

Составление расписаний в случае задач с неопределенными временами решения

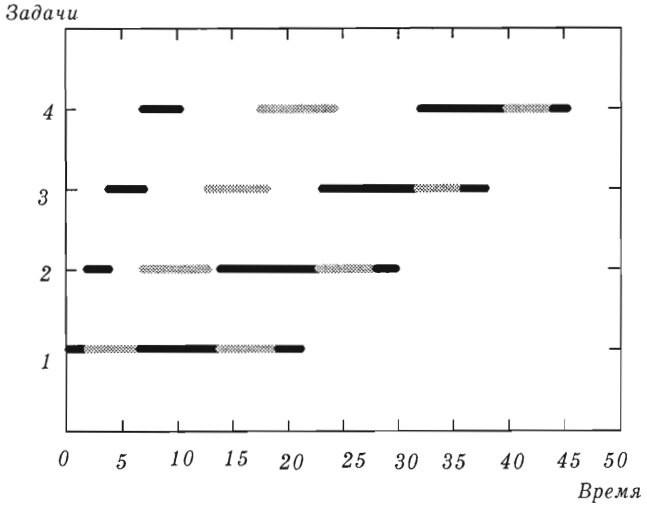
Рассмотрим теперь случай, когда времена решения задач известны приблизительно и задаются временны́ми интервалами $\tilde{\tau}_{ij} = [\underline{\tau}_{ij}, \overline{\tau}_{ij}]$ $i = \overline{1,m}$, $j = \overline{1,n}$, где $\underline{\tau}_{ij}$ и $\overline{\tau}_{ij}$ – нижняя и верхняя границы интервала $\tilde{\tau}_{ij}$ соответственно. Будем называть такую модель многопроцессорной системы недетерминированной системой, а детерминированную

■ *Таблица 1*

Задачи	Процессоры				
	1	2	3	4	5
1	2	5	7	5	2
2	2	6	9	5	1
3	3	5	9	4	2
4	3	6	8	4	1

■ *Таблица 2*

Суммы	Задачи			
	1	2	3	4
S_1	7	8	8	9
S_2	7	6	6	5



■ *Рис. 2. График Ганта для расписания из примера*

модель, получаемую из нее заменой интервальных времен $\tilde{\tau}_{ij} = [\underline{\tau}_{ij}, \overline{\tau}_{ij}]$ $i = \overline{1,m}$, $j = \overline{1,n}$ их нижними $\underline{\tau}_{ij}$ (верхними $\overline{\tau}_{ij}$) границами, – нижней (верхней) системой по отношению к исходной недерминированной системе.

Очевидно, что время $\tilde{T}(\pi)$ исполнения некоторого расписания π интервальной системой будет представлять собой временной интервал, который вычисляется как функция от значений временных интервалов $\tilde{\tau}_{ij} = [\underline{\tau}_{ij}, \overline{\tau}_{ij}]$ $i = \overline{1,m}$, $j = \overline{1,n}$, затрачиваемых на решение задач. При этом используются интервальные арифметические операции [5]:

$$\tilde{\tau}_i + \tilde{\tau}_j = [\underline{\tau}_i, \overline{\tau}_i] + [\underline{\tau}_j, \overline{\tau}_j] = [\underline{\tau}_i + \underline{\tau}_j, \overline{\tau}_i + \overline{\tau}_j];$$

$$\tilde{\tau}_i - \tilde{\tau}_j = [\underline{\tau}_i, \overline{\tau}_i] - [\underline{\tau}_j, \overline{\tau}_j] = [\underline{\tau}_i - \overline{\tau}_j, \overline{\tau}_i - \underline{\tau}_j];$$

$$\tilde{\tau}_i \tilde{\tau}_j = [\underline{\tau}_i, \overline{\tau}_i][\underline{\tau}_j, \overline{\tau}_j] =$$

$$= \left[\min\{\underline{\tau}_i\underline{\tau}_j, \underline{\tau}_i\overline{\tau}_j, \overline{\tau}_i\underline{\tau}_j, \overline{\tau}_i\overline{\tau}_j\},\ \max\{\underline{\tau}_i\underline{\tau}_j, \underline{\tau}_i\overline{\tau}_j, \overline{\tau}_i\underline{\tau}_j, \overline{\tau}_i\overline{\tau}_j\}\right];$$

$$\tilde{\tau}_i / \tilde{\tau}_j = [\underline{\tau}_i, \overline{\tau}_i] / [\underline{\tau}_j, \overline{\tau}_j] = [\underline{\tau}_i, \overline{\tau}_i][1/\overline{\tau}_j, 1/\underline{\tau}_j].$$

При поиске оптимального расписания необходимо сравнивать между собой интервальные значения времен выполнения различных вариантов расписаний. Для этого определим отношение частичного порядка на множестве интервалов.

Определение 2. Интервал $\tilde{\tau}_i = [\underline{\tau}_i, \overline{\tau}_i]$ не меньше интервала $\tilde{\tau}_j = [\underline{\tau}_j, \overline{\tau}_j]$ ($\tilde{\tau}_i \geq \tilde{\tau}_j$), если $\underline{\tau}_i \geq \underline{\tau}_j$ и $\overline{\tau}_i \geq \overline{\tau}_j$.

Известно [4], что время $\tilde{T}(\pi)$ исполнения некоторого расписания π недетерминированной системой равно интервалу $[\underline{T}(\pi), \overline{T}(\pi)]$, где $\underline{T}(\pi)$ и $\overline{T}(\pi)$ – время исполнения этого расписания соответственно для нижней и верхней систем. При этом оптимальное интервальное расписание образуется как пересечение множеств оптимальных расписаний для нижней и верхней систем и не всегда существует. Последнее происходит из-за того, что описания нижней и верхней систем слишком сильно расходятся, в результате чего им соответствуют разные оптимальные расписания. Если поиск оптимального расписания для недетерминированной системы представить как вычисление для каждого расписания времени его исполнения недетерминированной системой с последующим выбором расписания с минимальным временем исполнения, то об отсутствии решения станет известно при сопоставлении интервальных значений времен исполнения. В этом случае окажется, что полученные интервалы не сравнимы между собой. Таким образом, факт отсутствия оптимального решения является платой за недостаток информации.

Введем на множестве процессоров недетерминированной системы отношение доминирования «$\succ$», аналогичное отношению «$>$» для детерминированной системы.

Определение 3. Процессор L_q доминирует над процессором L_r ($L_q \succ L_r$), если $\min\limits_j \underline{\tau}_{q,j} \geq \max\limits_j \overline{\tau}_{r,j}$.

На основе этого отношения можно определить три базовых случая для недетеминированной системы, повторяющие случаи (1)–(3) с точностью до замены отношения «$>$» на отношение «$\succ$».

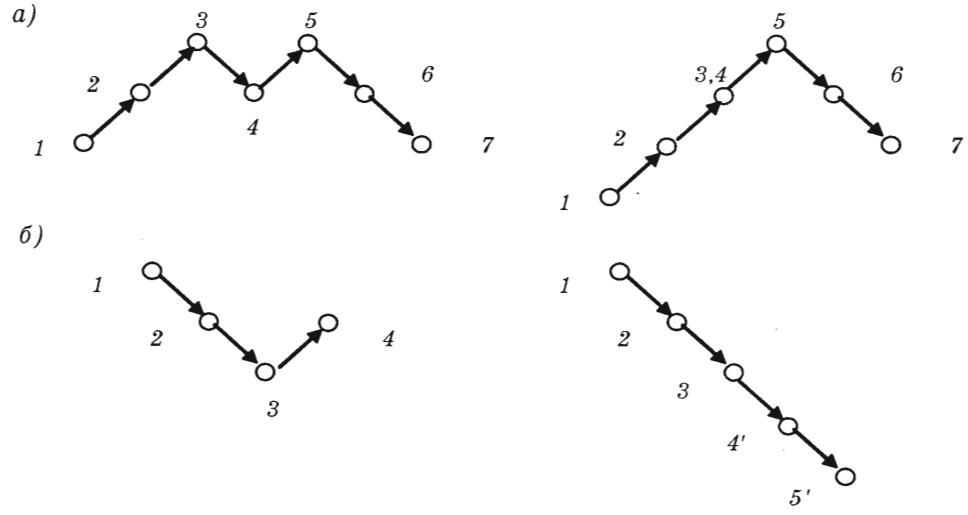
Предположим далее, что во всех рассматриваемых примерах существует оптимальное расписание, т. е. интервальные значения времен исполнения всех расписаний сравнимы между собой. Нетрудно заметить, что все рассуждения, приведенные в предыдущем разделе, а значит, и соотношения (4)–(6), справедливы для любой детерминированной системы, времена исполнения задач в которой принимают значения из соответствующих интервалов недетерминированной системы. На этом основании можно утверждать, что они остаются справедливыми и для недетерминированной системы при условии замены обычных арифметических операций на интервальные.

Приближенные решения с использованием базовых случаев

Как уже отмечалось во введении, в общем случае неизвестны алгоритмы поиска оптимальных расписаний полиномиальной сложности для конвейерной системы. В связи с этим предлагается использовать приближенные решения, предполагающие преобразование задачи к виду, удовлетворяющему одному из условий (1)–(3) для детерминированных систем или аналогичным условиям для недетерминированных систем. Это преобразование не является эквивалентным и в общем случае приводит к решению, отличному от оптимального. Рассмотрим два типа преобразований.

Сущность преобразования первого типа состоит в объединении двух или более последовательных фрагментов всех задач и соответственно в агрегировании исполняющих их процессоров, т. е. в замене двух или более процессоров одним с эквивалентной производительностью. При этом времена работы агрегированных процессоров определяются суммой времен работы объединяемых процессоров. Пример такого преобразования представлен на рис. 3, *а*. В этом примере между собой объединяются фрагменты 3 и 4, причем предполагается, что фрагмент 5 доминирует над объединенным фрагментом. В результате получаем задачу, соответствующую случаю 3.

Сущность преобразования второго типа состоит в расщеплении одного или более последовательных фрагментов всех задач на большее число фрагментов и соответственно в замене одного из процессоров двумя или более виртуальными процессорами. Пример такого преобразования представлен на рис. 3, *б*. В этом примере фрагмент 4 расщепляется на два фрагмента 4′ и 5′. При этом предполагается, что фрагмент 3 доминирует над получаемым в результате расщепления фрагментом 4′, а фраг-



■ *Рис. 3. Примеры преобразований задач*

мент 4′ – над фрагментом 5′. В результате получаем задачу, соответствующую случаю 1. Очевидно, что такое расщепление с точностью до числа образуемых при расщеплении фрагментов всегда возможно.

Каждое из этих преобразований можно охарактеризовать достаточно очевидным, но важным свойством. Для первого преобразования (объединение фрагментов) оно звучит следующим образом.

Свойство 1. Значение времени $\overline{T}_{\text{опт}}(m',n)$ $(m' < m)$ исполнения оптимального расписания, получаемого для преобразованной задачи, является верхней границей для значения времени $T_{\text{опт}}(m,n)$ исполнения оптимального расписания для исходной задачи.

Действительно, в результате первого преобразования параллелизм решения для заданной совокупности задач снижается, а значит, время решения всех задач не может уменьшиться.

Для второго преобразования (расщепление фрагментов) свойство звучит следующим образом.

Свойство 2. Значение времени $\underline{T}_{\text{опт}}(m'',n)$ $(m'' > m)$ исполнения оптимального расписания, получаемого для преобразованной задачи, является нижней границей для значения времени $T_{\text{опт}}(m,n)$ исполнения оптимального расписания для исходной задачи.

Действительно, в результате второго преобразования параллелизм решения для заданной совокупности задач возрастает, а значит, время решения всех задач не может уменьшиться.

Этими свойствами можно руководствоваться при определении правил поиска приближенных решений, а именно, выбирать в качестве решения задачи расписание, оптимальное с точки зрения верхней или нижней границ. При решении задачи может оказаться возможным определение более, чем одной границы. Если в этом случае разные границы предъявляют к расписанию непротиворечивые требования, то они могут быть удовлетворены одновременно.

Заключение

В настоящей статье исследованы вопросы построения расписаний для конвейерных вычислительных систем. При этом проанализированы три базовых случая, для которых указаны по существу беспереборные алгоритмы построения оптимальных расписаний для произвольного числа задач и процессоров. Эти случаи проанализированы как при условии, что времена решения задач точно известны, так и при условии, что эти времена заданы интервалами. Дополнительно рассмотрена возможность получения приближенных решений путем сведения исходной задачи к рассмотренным базовым случаям.

Литература

1. **Конвей Р. В., Максвелл В. Л., Миллер Л. В.** Теория расписаний. – М.: Наука, 1975. – 282 с.
2. **Левин В. И.** Структурно-логические методы исследования сложных систем с применением ЭВМ. – М.: Наука, 1987. – 304 с.
3. **Танаев В. С., Сотсков Ю.Н., Струсевич В.А.** Теория расписаний. Многостадийные системы. – М.: Наука, 1989. – 322 с.
4. **Левин В. И.** Оптимизация расписаний в М-стадийной системе с неопределенными временами обработки // Автоматика и телемеханика. – 2002. – № 2. – С. 129–136.
5. **Калмыков С. А., Шокин Ю. А., Юлдашев З. Х.** Методы интервального анализа. – Новосибирск: Наука, – 1986. – 294 с.