УДК 519.687.5:004.451.44

ПЛАНИРОВАНИЕ ЗАДАЧ В СИСТЕМАХ АВТОМАТИЗАЦИИ И УПРАВЛЕНИЯ ПРИ ЛИНЕЙНЫХ ИНТЕРВАЛЬНЫХ ОГРАНИЧЕНИЯХ РЕАЛЬНОГО ВРЕМЕНИ

М.В. Кавалеров, Н.Н. Матушкин

Пермский государственный технический университет

Предложен ряд алгоритмов, обеспечивающих назначение параметров задач реального времени для планирования с фиксированными приоритетами при любых линейных интервальных ограничениях реального времени. Показано, что предложенные алгоритмы позволяют повысить эффективность планирования и, как следствие, реализовать при имеющихся вычислительных ресурсах более качественное управление.

ВВЕДЕНИЕ

В системах автоматизации и управления (САиУ) отдельное вычислительное устройство часто выполняет несколько задач реального времени (РВ). Каждой такой задаче соответствует свое ограничение РВ, которое накладывается на характеристики процесса выполнения запросов, формируемых этой задачей. При этом необходимо планировать множество задач, т. е. разделение процессорного времени между запросами различных задач при условии соблюдения ограничений РВ. Более эффективное планирование позволяет соблюдать более жесткие ограничения при данных аппаратно-программных средствах. Например, реализуется более строгая периодичность опроса датчиков, уменьшаются задержки при формировании управляющих воздействий. Зачастую это означает повышение качества управления. Более эффективное планирование также снижает требования к быстродействию вычислительного устройства, что обеспечивает снижение затрат и уменьшение срока окупаемости САиУ. Многие исследования посвящены проблемам планирования задач РВ. Хороший обзор истории и состояния этих исследований дан в работе [1].

Так, широкое распространение получила модель планирования с фиксированными приоритетами (ПФП) — Fixed Priority Scheduling, обеспечивающая компромисс между предсказуемостью и гибкостью. В стандартной модели ПФП применяются только ограничения РВ, выражаемые через начальное смещение, период и крайний срок. Эти ограничения будем называть *стандартными* ограничениями (СО). Многие исходные ограничения РВ представляют собой *нестандартные* ограничения (НО). Отсюда возникает проблема планирования при НО в условиях ПФП, которая формулируется так. Требуется установить параметры каждой задачи РВ (начальное смещение, период, приоритет) так, чтобы при выполнении задач в условиях ПФП гарантированно соблюдались все СО и НО, или сообщить о невозможности гарантирования.

Рассмотрим следующий пример. Пусть на однопроцессорном контроллере выполняются две задачи РВ (τ_1, τ_2) для двух независимых контуров управления (рис. 1, *а*). Каждая *i*-я задача должна периодически формировать запросы ($\tau_{i,1}$, $\tau_{i,2}$, ...), и для выполнения каждого запроса требуется процессорное время выполнения (здесь оно считается постоянным) для формирования очередного воздействия на объект (рис. 1, *б*). В случае отдельного процессора для каждой задачи проблем не возникает, и они выполняются, как показано на рис. 1, *б*. Однако при общем процессоре возникает взаимовлияние. Пусть начальные смещения (O_1, O_2) и периоды (T_1, T_2) менять нельзя, тогда в рамках ПФП остается лишь менять приоритеты (π_1, π_2).

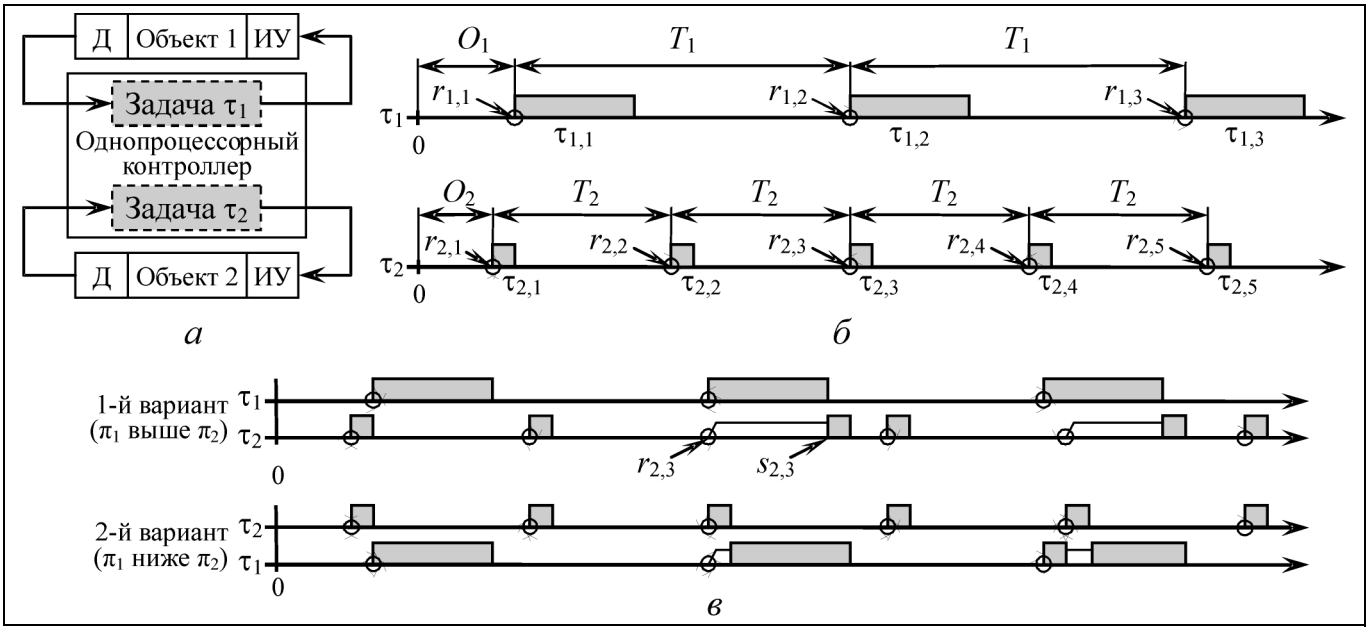

Рис. 1. Пример планирования задач РВ:
а — реализация двух контуров управления с использованием одного контроллера и двух задач РВ; *б* — временные диаграммы выполнения этих задач в идеальном случае (предполагается отсутствие влияния соседней задачи); *в* — два варианта выполнения задач, соответствующие двум вариантам планирования (здесь задачи делят между собой время одного процессора); *Д* — датчик; *ИУ* — исполнительное устройство

Поэтому существуют только два варианта планирования: π_1 выше π_2 и π_1 ниже π_2, приводящие к двум вариантам выполнения задач (рис. 1, *в*). Согласно ПФП запрос задачи с более высоким приоритетом прерывает выполнение запроса задачи с более низким приоритетом. На рисунке кружками отмечены моменты $r_{i,\,v}$ появления запросов $\tau_{i,\,v}$ в очереди запросов, а прерывание запроса обозначено линией над соответствующей временной осью. При этом первый из указанных вариантов приводит к значительному нарушению строгой периодичности для задачи τ_2 ($r_{2,\,3} \neq s_{2,\,3}$, где $s_{2,\,3}$ — начало выполнения запроса $\tau_{2,\,3}$). Простое изменение приоритетов, приводящее ко второму варианту, обеспечивает строгую периодичность для задачи τ_2, и небольшое нарушение периодичности для задачи τ_1. Пусть такое небольшое нарушение для задачи τ_1 оказывается допустимым. Тогда решением проблемы планирования будет второй вариант с соответствующими значениями (O_1, T_1, π_1) и (O_2, T_2, π_2).

Однако в приведенном примере имеется всего два варианта планирования, и из них легко выбрать подходящий. Возможность изменения значений O_i и T_i в заданных диапазонах, а также наличие $n!$ вариантов назначения приоритетов для n задач приводят к тому, что полный перебор вариантов становится практически нереализуемым. Известно, что указанная проблема планирования является NP-трудной. Поэтому естественный подход к ее решению — это разработка эффективных эвристических алгоритмов, которые за приемлемое время с высокой вероятностью находят значения $\{O_i,\ T_i,\ \pi_i \mid i \in [1,\ n]\}$, обеспечивающие подходящий вариант планирования. Сложность разработки таких алгоритмов зависит от видов НО, т. е. от *целевого класса* НО.

Подходы к решению указанной проблемы представлены в работах [2—4]. В работах [2, 3] предложены решения для таких НО, как ограничения предшествования и ограничения, представимые в виде последовательности разрешенных интервалов («target windows» или «valid scopes»). Однако многие НО могут быть заданы в виде указанной последовательности только путем упрощений, приводящих к снижению общей эффективности из-за искусственного ужесточения ограничений. В работе [4] предложен подход, потенциально ориентированный на наиболее широкий целевой класс НО. Он основан на применении генетического алгоритма, к его недостаткам можно отнести неизменность периодов задач, а также очевидные свойства, связанные с его недетерминированностью. Подобный подход выгодно применять, когда детерминированный эвристический алгоритм не может гарантировать соблюдение СО и НО.

В настоящей работе предлагаются детерминированные эвристические алгоритмы, обеспечивающие решение указанной проблемы планирования для целевого класса НО, часто встречающихся на практике, но не входящих в целевые классы НО известных подходов.

1. МОДЕЛЬ ПЛАНИРОВАНИЯ С ФИКСИРОВАННЫМИ ПРИОРИТЕТАМИ

Планирование — однопроцессорное. Есть множество задач $\{\tau_i\} = \{\tau_1, ..., \tau_n\}$ жесткого РВ. Каждая задача τ_i формирует запросы, каждый v-й запрос $\tau_{i,v}$ формируется в момент времени $r_{i,v}$. Для любой *периодической* задачи τ_i при $\forall\ v \geqslant 1$ всегда $r_{i,v} = O_i + (v-1)T_i$, где O_i — начальное смещение; T_i — период. Задача τ_i может иметь СО в виде условия $f_{i,v} \leqslant r_{i,v} + D_i$, где $f_{i,v}$ — время завершения выполнения $\tau_{i,v}$, D_i — крайний срок завершения выполнения каждого запроса задачи τ_i, исчисляемый от момента появления соответствующего запроса. Стандартное ограничение периодической задачи τ_i удобнее представлять в виде условия [5]

$$\begin{cases} s_{i,v} \geq O_i^* + (v-1)T_i^*, \\ f_{i,v} \leq O_i^* + (v-1)T_i^* + D_i, \end{cases} \qquad (1)$$

где $s_{i,v}$ — время начала выполнения задачи $\tau_{i,v}$; O_i^*, T_i^* — параметры СО, $O_i^* \geqslant 0$, $T_i^* > 0$. Здесь и далее предполагается, что условие, задающее ограничение РВ, должно соблюдаться при $\forall v \geqslant 1$. В работе [5] показано, что в случае СО периодической задачи надо устанавливать $O_i = O_i^*$ и $T_i = T_i^*$. При выполнении запроса $\tau_{i,v}$, кроме моментов $s_{i,v}$ и $f_{i,v}$, можно выделить еще два значимых момента времени $x_{i,v}$ и $y_{i,v}$, при этом $s_{i,v} \leqslant x_{i,v} < y_{i,v} \leqslant f_{i,v}$. Например, в момент $x_{i,v}$ происходит получение информации, а в момент $y_{i,v}$ — формирование управляющего воздействия. Тогда с запросом $\tau_{i,v}$ связано четыре момента времени: $s_{i,v}$, $x_{i,v}$, $y_{i,v}$, $f_{i,v}$, и существует шесть различных компонентов запроса $\tau_{i,v}$ с соответствующими длительностями (рис. 2). Для каждого компонента определяются нижняя и верхняя оценка длительности его выполнения. Поэтому считается, что известны 12 оценок: Csx_i^{Lo}, Csx_i^{Up}, Csy_i^{Lo}, Csy_i^{Up}, Csf_i^{Lo}, Csf_i^{Up}, Cxy_i^{Lo}, Cxy_i^{Up}, Cxf_i^{Lo}, Cxf_i^{Up}, Cyf_i^{Lo} и Cyf_i^{Up}, где Csx_i^{Lo} и Csx_i^{Up} — нижняя и верхняя оценки длины интервала $[s_{i,v}, x_{i,v}]$ по $\forall v \geqslant 1$ при отсутствии прерываний запроса $\tau_{i,v}$; другие оценки определяются аналогично. Преимущества разделения запроса на компоненты указаны в работе [6]. Каждая задача τ_i имеет уникальный фиксированный приоритет π_i в виде целого числа из интервала $[1, n]$. При этом задача с самым высоким приоритетом имеет приоритет, равный 1.

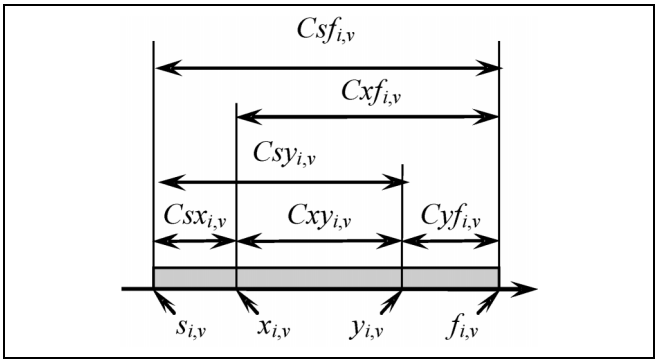

Рис. 2. Длительности компонентов запроса $\tau_{i,v}$

В условиях ПФП можно выделить две стадии: планирование (выполняется до старта САиУ) и диспетчеризация (выполняется во время функционирования САиУ). Диспетчеризация предполагает обслуживание очереди запросов, формируемых задачами РВ, с учетом фиксированных приоритетов соответствующих задач. Пример двух вариантов процесса диспетчеризации показан на рис. 1, *в*. Выполнение запросов периодической задачи τ_i зависит только от значений O_i, T_i, π_i и текущего состояния очереди. Множество $\{O_i, T_i, \pi_i | i \in [1, n]\}$ порождает возможные варианты выполнения запросов задач $\{\tau_i\}$. Если в случае каждого из этих вариантов соблюдается ограничение РВ задачи τ_i, то соблюдение этого ограничения *гарантируется*, и задача τ_i называется *выполнимой*. Множество задач $\{\tau_i\}$ называется *выполнимым*, если обеспечивается выполнимость каждой задачи из этого множества и гарантируется, что размер очереди ограничен сверху. До старта системы надо обеспечить выполнимость множества задач $\{\tau_i\}$, указав подходящее множество $\{O_i, T_i, \pi_i | i \in [1, n]\}$, или сообщить, что это сделать не удается. Изложенное и есть описание проблемы, ранее сформулированной во Введении. Именно такая проблема решается на стадии планирования.

2. КЛАСС ЛИНЕЙНЫХ ИНТЕРВАЛЬНЫХ ОГРАНИЧЕНИЙ

В работе [7] дано определение обобщенного НО. В классе обобщенных НО предлагается выделить более узкий класс, для которого становит-

ся возможным разработать универсальный алгоритм планирования и который будет целевым классом НО.

Линейное интервальное ограничение (ЛИО) для задачи τ_i, обозначаемое Ξ_i, — это ограничение, представимое при $\forall\ v \geqslant 1$ в виде условия

$$\begin{cases} x_{i,\,v} \geq x_{i,\,v}^{\min} = \max(x_{i,\,v,\,1}^{\min}, ..., x_{i,\,v,\,w_1}^{\min}), \\ x_{i,\,v} \leq x_{i,\,v}^{\max} = \min(x_{i,\,v,\,1}^{\max}, ..., x_{i,\,v,\,w_2}^{\max}), \\ y_{i,\,v} \geq y_{i,\,v}^{\min} = \max(y_{i,\,v,\,1}^{\min}, ..., y_{i,\,v,\,w_3}^{\min}), \\ y_{i,\,v} \leq y_{i,\,v}^{\max} = \min(y_{i,\,v,\,1}^{\max}, ..., y_{i,\,v,\,w_4}^{\max}), \\ xy_{i,\,v} \geq xy_{i,\,v}^{\min} = \max(xy_{i,\,v,\,1}^{\min}, ..., xy_{i,\,v,\,w_5}^{\min}), \\ xy_{i,\,v} \leq xy_{i,\,v}^{\max} = \min(xy_{i,\,v,\,1}^{\max}, ..., xy_{i,\,v,\,w_6}^{\max}), \end{cases}$$

где $xy_{i,\,v}$ — длина интервала $[x_{i,\,v},\ y_{i,\,v}]$; $x_{i,\,v}^{\min}$, $x_{i,\,v}^{\max}$, $y_{i,\,v}^{\min}$, $y_{i,\,v}^{\max}$, $xy_{i,\,v}^{\min}$ и $xy_{i,\,v}^{\max}$ — минимальные и максимальные значения интервалов допустимых значений $x_{i,\,v}$, $y_{i,\,v}$ и $xy_{i,\,v}$ соответственно; w_1, ..., w_6 задают число вариантов; $x_{i,\,v,\,w}^{\min}$, $x_{i,\,v,\,w}^{\max}$, $y_{i,\,v,\,w}^{\min}$, $y_{i,\,v,\,w}^{\max}$, $xy_{i,\,v,\,w}^{\min}$ и $xy_{i,\,v,\,w}^{\max}$ обозначают w-й вариант соответствующих значений, и каждый из вариантов — это или $-\infty$ (может быть только после знака «$\geqslant$»), или ∞ (может быть только после знака «$\leqslant$»), или выражение, представимое в виде

$$\sum_{k=1}^{k_\alpha^{\max}} \alpha_k x_{i,\,v-k} + \sum_{k=1}^{k_\beta^{\max}} \beta_k y_{i,\,v-k} + \gamma v + \delta, \qquad (2)$$

где $k_\alpha^{\max}$, $k_\beta^{\max}$ — целые неотрицательные значения; α_k при $\forall k \in [1,\ k_\alpha^{\max}]$, β_k при $\forall k \in [1,\ k_\beta^{\max}]$, γ, δ — константы, заданные для данного варианта; при этом все $x_{i,\,v-k}$, $y_{i,\,v-k}$ для $v - k \leqslant 0$, встречающиеся в выражениях вида (2), являются константами, заданными для ограничения Ξ_i.

Множество всех возможных ЛИО формирует класс ЛИО, который, в свою очередь, входит в класс обобщенных НО, приведенный в работе [7].

3. ПРИМЕРЫ ЛИНЕЙНЫХ ИНТЕРВАЛЬНЫХ ОГРАНИЧЕНИЙ

К классу ЛИО относится НО задачи контура управления (ЗКУ), задаваемое условием

$$\begin{cases} T_i^{xx\min} \leq x_{i,\,v} - x_{i,\,v-1} \leq T_i^{xx\max}, \\ y_{i,\,v} - x_{i,\,v} \leq T_i^{xy\max}, \end{cases} \qquad (3)$$

где $T_i^{xx\min}$, $T_i^{xx\max}$ и $T_i^{xy\max}$ — константы; при этом заранее устанавливается значение $x_{i,\,0}$. Это НО является обобщением ограничения, определенного в работе [8, п. 5.2], при условии, что измерение и управляющее воздействие осуществляются одной задачей. Обобщение состоит в использовании величин $x_{i,\,v}$ и $y_{i,\,v}$ вместо $s_{i,\,v}$ и $f_{i,\,v}$, естественно, после перехода к обозначениям, применяемым здесь. Можно считать, что обе задачи (см. рис. 1) имеют НО (3). При этом $x_{i,\,v} = s_{i,\,v}$, $y_{i,\,v} = f_{i,\,v}$ для $\forall i \in [1, 2]$ и $\forall v \geqslant 1$, а также $T_1^{xx\min} = T_1^{xx\max} - \Delta T_1$, $T_2^{xx\min} = T_2^{xx\max}$, $T_1^{xy\max} > T_1^{xx\max}$, $T_2^{xy\max} > T_2^{xx\max}$, где ΔT_1 определяет степень допустимого нарушения периодичности для задачи τ_1.

Далее для краткости указываются только некоторые наиболее простые примеры ЛИО без подробного описания каждого из них.

К классу ЛИО относится такое часто встречающееся НО, как, например, ограничение, обозначенное как Dynamic instance timing constraints в работе [9, п. 3]. Это ограничение можно обобщить аналогично тому, как это было сделано для НО ЗКУ. В этом случае, как и для НО (3), будут более точно учитываться реальные особенности выполнения задач РВ [6].

Важно, что различные практически значимые модификации рассмотренных ограничений также представляют собой ЛИО. Например, НО (3) содержит условие, ограничивающее значение $x_{i,\,v} - x_{i,\,v-1}$. Можно рассмотреть аналогичные условия, ограничивающие значения $y_{i,\,v} - y_{i,\,v-1}$ и $y_{i,\,v} - x_{i,\,v-1}$. Различные комбинации этих условий задают новые виды практически значимых ограничений, относящихся к классу ЛИО. В рассматриваемое ограничение также можно добавить условие, ограничивающее значение $x_{i,\,v} - x_{i,\,v-m}$, где m — константа, $m > 1$. Тогда ограничение на значение $x_{i,\,v} - x_{i,\,v-1}$ может быть ослаблено путем надлежащего подбора допустимого диапазона для значения $x_{i,\,v} - x_{i,\,v-m}$, что ограничит период для моментов $x_{i,\,v}$, усредненный по m интервалам. Такое уточнение ограничения способствует повышению эффективности планирования, и полученное ограничение также будет принадлежать классу ЛИО. Можно сделать подобные уточнения для значений $y_{i,\,v} - y_{i,\,v-1}$ и $y_{i,\,v} - x_{i,\,v-1}$ и добавить условия для других значений m. Все это указывает на большое число практически значимых видов НО, принадлежащих классу ЛИО.

4. БАЗОВЫЙ АЛГОРИТМ ПЛАНИРОВАНИЯ

Традиционный подход к планированию при наличии НО состоит в преобразовании нестандартного ограничения в стандартное и последующем применении известных методов для стандартной модели планирования [1]. Такой подход будем называть *базовым*. Например, можно привести работы [9, 10], в которых используется базовый подход. Преобразование НО в СО состоит в получении СО, соблюдение которого гарантирует соблюдение данного НО. Такое СО (и сочетание (O_i^*, T_i^*, D_i), полностью определяющее данное СО) будем называть *допустимым*.

Рассмотрим пример. Пусть задача τ_i имеет НО (3), при этом $x_{i,\,v} = s_{i,\,v}$, $y_{i,\,v} = f_{i,\,v}$ для $\forall v \geqslant 1$. На рис. 3, *а* показаны значения $O_i^* = O_i = 0$ и $T_i^* = T_i$, D_i, соответствующие допустимому СО для этого НО. Действительно, пусть гарантируется соблюдение этого СО, тогда, учитывая условие (1), каждый запрос выполняется только внутри соответствующего интервала длиной D_i. При этом наибольшее значение $s_{i,\,v} - s_{i,\,v-1}$ (и соответственно $x_{i,\,v} - x_{i,\,v-1}$) достигается, когда запрос $\tau_{i,\,v-1}$ «прижимается» к левой границе интервала, а запрос $\tau_{i,\,v}$ — к правой границе (например, см. запросы $\tau_{i,\,1}$, $\tau_{i,\,2}$ на рис. 3, *а*). Значение $T_i^{xx\max}$ для НО (3) показано на рис. 3, поэтому очевидно, что при соблюдении СО с данными значениями T_i^* и D_i гарантировано соблюдение условия $x_{i,\,v} - x_{i,\,v-1} \leqslant T_i^{xx\max}$ для $\forall v > 1$. Аналогично можно рассмотреть $T_i^{xx\min}$ и запросы $\tau_{i,\,2}$ и $\tau_{i,\,3}$ (см. рис. 3, *а*), реализующие ситуацию с наименьшим значением $x_{i,\,v} - x_{i,\,v-1}$, отсюда очевидно гарантированное соблюдение условия $T_i^{xx\min} \leqslant x_{i,\,v} - x_{i,\,v-1}$ для $\forall v > 1$. Тем самым, гарантируется соблюдение неравенств в первой строке НО (3) для $\forall v > 1$. Остается проверить эту строку при $v = 1$. Пусть оказалось, что значение $x_{i,\,0}$ (параметр НО ЗКУ) равняется T_i^*, тогда, очевидно, при $O_i^* = O_i = 0$ гарантируется соблюдение неравенств в первой строке при $v = 1$. При этом выполнение неравенства во второй строке НО (3) гарантируется при $\forall\ v \geqslant 1$ тем, что $D_i = T_i^{xy\max}$ (см. рис. 3). Таким образом, соблюдение данного СО гарантирует соблюдение НО (3), т. е. СО является допустимым.

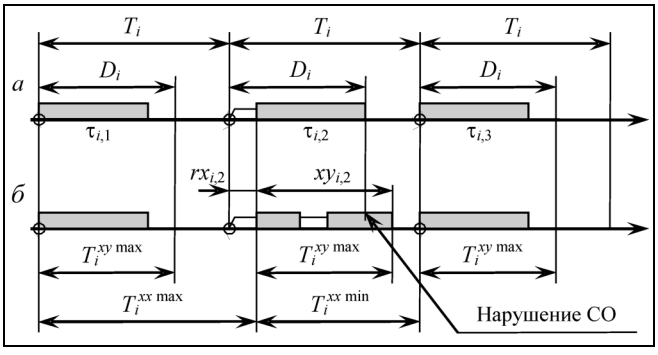

Рис. 3. Примеры выполнения задачи τ_i с нестандартным ограничением вида (3) и допустимым стандартным ограничением: *а* — первый вариант; *б* — второй вариант

Подобным образом, рассматривая крайние ситуации, можно преобразовать заданное ЛИО в допустимое СО. Более того, в работе [11] предложен алгоритм, реализующий это для любого ЛИО. Тогда очевиден базовый алгоритм планирования для целевого класса ЛИО.

Алгоритм 1. Базовый алгоритм планирования при любых ЛИО.

1. Для каждой задачи τ_i, имеющей ограничение Ξ_i, вызывается алгоритм 3 из работы [11], который реализует преобразование ЛИО Ξ_i в СО. Если этот алгоритм завершается с сообщением, что задача τ_i может не выполняться, тогда эта задача исключается из множества $\{\tau_i\}$. Если этот алгоритм завершается с сообщением, что допустимое СО сформировать не удалось, то определяемый алгоритм завершается с сообщением о невозможности обеспечить выполнимость множества задач $\{\tau_i\}$.

2. Для каждой периодической задачи τ_i устанавливается $O_i = O_i^*$, $T_i = T_i^*$.

3. Выполняется планирование при наличии только СО согласно алгоритму 1 из работы [12]. После этого определяемый алгоритм завершается.♦

5. АЛГОРИТМ А

Еще в работе [9] было отмечено, что преобразование нестандартного ограничения в стандартное приводит к снижению эффективности планирования из-за практически неизбежного ужесточения получаемого СО. В случае НО (3) пример такой ситуации показан на рис. 3, *б*, где иллюстрируется выполнение той же задачи, что и на рис. 3, *а*. Отличие только в том, что запрос $\tau_{i,\,2}$ прерывается запросом задачи с более высоким приоритетом. На рис. 3 не показаны задачи с более высокими приоритетами, но их присутствие проявляется также в задержке начала выполнения запроса $\tau_{i,\,2}$. Указанное прерывание приводит к выходу запроса

$\tau_{i,2}$ за интервал длиной D_i, т. е. допустимое СО нарушается. Однако исходное НО ЗКУ соблюдается, так как момент $x_{i,2}$ не меняется (по сравнению с рис. 3, *а*), что обеспечивает выполнение неравенств в первой строке условия (3), а значение $y_{i,2} - x_{i,2}$ не превышает величины $T_i^{xy\max}$, что обеспечивает выполнение неравенства во второй строке условия (3). Тем самым складывается ситуация, когда из нарушения допустимого СО не следует нарушение исходного НО. В такой ситуации базовый алгоритм планирования при выполнении шага 3 «придет» к выводу, что допустимое СО для задачи τ_i может нарушаться, и поэтому невозможно обеспечить выполнимость множества задач $\{\tau_i\}$. Однако выполнимость множества $\{\tau_i\}$ обеспечивается, так как исходное НО ЗКУ соблюдается, как указано ранее. Ведь надо соблюдать именно исходное НО, а не СО, полученное на основе этого НО. Таким образом, базовый алгоритм выдает излишне пессимистичный результат, и на его основе на этапе проектирования может быть принято решение о покупке более быстродействующего и дорогого вычислительного устройства. Однако таких затрат можно избежать, если применить более эффективный алгоритм планирования, который непосредственно проверяет соблюдение исходных НО, не прибегая к посредничеству допустимых СО, как это делается в случае базового алгоритма.

Для выполнения такой непосредственной проверки надо оценивать задержки при выполнении запросов. Так, на рис. 3, *б* задержка от момента $r_{i,2}$ до момента $x_{i,2}$ обозначена как $rx_{i,2}$. Пусть rx_i^{Lo}, rx_i^{Up}, ry_i^{Lo}, ry_i^{Up}, xy_i^{Lo} и xy_i^{Up} обозначают соответственно нижние и верхние оценки длин интервалов $[r_{i,v}, x_{i,v}]$, $[r_{i,v}, y_{i,v}]$, $[x_{i,v}, y_{i,v}]$ по $\forall v \geqslant 1$. Пусть оказалось, что $rx_i^{Lo} = 0$ и $rx_i^{Up} = rx_{i,2}$, тогда все возможные значения $rx_{i,v}$ лежат в интервале $[0, rx_{i,2}]$. Тогда очевиден вывод, что неравенства в первой строке НО (3) соблюдаются, так как наиболее критичная из возможных ситуаций показана как раз на рис. 3, *б*. Аналогично, пусть оказалось, что $xy_i^{Up} = xy_{i,2}$, тогда очевидно, что соблюдаются неравенства во второй строке НО (3). Таким образом, на основе указанных оценок можно определить выполнимость задачи τ_i. К примеру, если оказывается, что $xy_i^{Up} > xy_{i,2}$, то ясно, что задача τ_i невыполнима. Вообще, используя рис. 3, нетрудно сформировать условие выполнимости для задачи с НО (3) в общем виде, и это будет условие

$$\begin{cases} O_i \in [(x_{i,0} + T_i^{xx\min} - rx_i^{Lo}), (x_{i,0} + T_i^{xx\max} - rx_i^{Up})], \\ T_i \in [(T_i^{xx\min} - rx_i^{Lo} + rx_i^{Up}), (T_i^{xx\max} + rx_i^{Lo} - rx_i^{Up})], \\ xy_i^{Up} \le T_i^{xy\max}. \end{cases} \quad (4)$$

Если условие (4) выполняется, то задача с НО ЗКУ выполнима, т. е. гарантируется соблюдение этого ограничения. При этом все особенности влияния задач с более высокими приоритетами учитываются с помощью оценок rx_i^{Lo}, rx_i^{Up} и xy_i^{Up}. Однако для других ЛИО надо находить свои условия выполнимости. Для решения этой проблемы предлагается следующий алгоритм.

Алгоритм 2. Формирование условия выполнимости задачи τ_i в случае любого ЛИО.

Данный алгоритм можно описать как алгоритм, получаемый из алгоритма 1, приведенного в работе [11], на основе следующих его модификаций. В ходе шага 1 подставляются не выражения, указанные в описании этого шага, а соответственно выражения $O_i + (v-1)T_i + rx_i^{Lo}$, $O_i + (v-1)T_i + rx_i^{Up}$, $O_i + (v-1)T_i + ry_i^{Lo}$, $O_i + (v-1)T_i + ry_i^{Up}$, xy_i^{Lo}, xy_i^{Up}. При этом если в ходе шага 1 удаляются все неравенства, то алгоритм завершается с результатом в виде всюду истинного условия выполнимости. Далее, в ходе шага 4 в качестве нижней и верхней оценки $x_{i,z-k}$, нижней и верхней оценки $y_{i,z-k}$ следует использовать соответственно величины $O_i + (v-k-1)T_i + rx_i^{Lo}$, $O_i + (v-k-1)T_i + rx_i^{Up}$, $O_i + (v-k-1)T_i + ry_i^{Lo}$ и $O_i + (v-k-1)T_i + ry_i^{Up}$. Наконец, в ходе шага 8 указанным образом формируется не условие допустимости (O_i^*, T_i^*, D_i), а условие выполнимости задачи τ_i.♦

В получаемом условии выполнимости могут содержаться любые из оценок rx_i^{Lo}, rx_i^{Up}, ry_i^{Lo}, ry_i^{Up}, xy_i^{Lo} и xy_i^{Up}, поэтому надо уметь их вычислять. В работе [13] предложен алгоритм вычисления этих оценок и доказательство корректности вычисляемых оценок.

По определению выполнимости множества задач $\{\tau_i\}$, надо обеспечивать ограниченность очереди запросов. При наличии только СО ограниченность очереди следует из гарантированного соблюдения СО. В общем случае надо отдельно проверять ограниченность размера очереди запро-

сов. Пусть U_τ^{Up} обозначает верхнюю оценку загрузки задачами $\{\tau_i\}$, вычисляемую как

$$U_\tau^{Up} = \sum_{i=1}^{n} \frac{Csf_i^{Up}}{T_i} 100\ \%. \qquad (5)$$

Очевидно, размер очереди запросов будет ограничен сверху, если $U_\tau^{Up} \leqslant 100\ \%$.

Учитывая возможность проверки условия выполнимости для любого ЛИО, а также условие $U_\tau^{Up} \leqslant 100\ \%$, предлагается реализовать алгоритм планирования на основе непосредственного анализа выполнимости задач, имеющих ЛИО. Для краткости такой алгоритм будет называть *алгоритмом А*, при этом «А» соответствует первой букве слова «анализ».

Алгоритм 3. Алгоритм А, реализующий планирование при любых ЛИО.

1. Последовательно выполняются все действия шагов 1 и 2 алгоритма 1.

2. Если $U_\tau^{Up} > 100\ \%$, то алгоритм завершается с сообщением о невозможности обеспечить выполнимость задач $\{\tau_i\}$.

3. Для каждой задачи τ_i, имеющей ЛИО Ξ_i, с помощью алгоритма 2 формируется условие выполнимости и с помощью алгоритма 3 из работы [13], для которого задается $\beta = 1$ в составе его входных данных, вычисляются те из оценок rx_i^{Lo}, rx_i^{Up}, ry_i^{Lo}, ry_i^{Up}, xy_i^{Lo} и xy_i^{Up}, которые встречаются в сформированном условии выполнимости.

4. Выполняется алгоритм 1 из работы [12] с функцией feasible, реализованной так: если τ_i имеет ЛИО, то результат функции — это значение истинности условия выполнимости, полученного для задачи τ_i в ходе шага 3; иначе результат функции — это значение истинности условия $rf_i^{Up} \leqslant D_i$, где rf_i^{Up} — верхняя оценка длины $[r_{i,v}, f_{i,v}]$ по $\forall v \geqslant 1$. Алгоритм завершается. ♦

6. АЛГОРИТМ АП

В случае алгоритма А значение T_i для задачи с ограничением Ξ_i определяется через допустимое СО (см. шаг 1 алгоритма 3 и шаг 2 алгоритма 1) без учета выполнимости задачи. Поэтому могут быть пропущены более подходящие варианты T_i, что снижает эффективность планирования.

Рассмотрим пример. Пусть $\{\tau_i\} = \{\tau_1, \tau_2\}$. Задачи τ_1 и τ_2 являются периодическими и имеют НО (3). При этом $x_{i,v} = s_{i,v}$, $y_{i,v} = f_{i,v}$ для $\forall i \in \{1, 2\}$ и $\forall v \geqslant 1$, а также $Csf_1^{Lo} = Csf_1^{Up} = 5$, $Csf_2^{Lo} = Csf_2^{Up} = 10$. Параметры НО (3) для обеих задач: $T_1^{xx\min} = 5$, $T_1^{xx\max} = 15$, $T_1^{xy\max} = 15$, $x_{1,0} = s_{1,0} = -10$, $T_2^{xx\min} = 10$, $T_2^{xx\max} = 20$, $T_2^{xy\max} = 15$, $x_{2,0} = s_{2,0} = -10$. Пусть $\theta = 1$, где θ определяет желаемое значение D_i/T_i для всех задач с ЛИО (подробнее параметр θ рассматривается в работе [11]). Тогда выполнение алгоритма А (см. шаг 1) приводит к тому, что $T_1 = 10$, $T_2 = 15$. При этом $U_\tau^{Up} \approx 117\ \%$, и алгоритм А не сможет обеспечить выполнимость задач $\{\tau_i\}$ (см. шаг 2 алгоритма А). При других значениях θ алгоритм А также не обеспечивает их выполнимость либо по аналогичной причине ($U_\tau^{Up} > 100\ \%$), либо потому, что $T_2 \neq 15$, ведь при $T_2 \neq 15$ не соблюдается выражение во второй строке условия (4), учитывая, что $rx_2^{Lo} = 0$, $rx_2^{Up} = 5$. Получается, что алгоритм А в принципе не может обеспечить выполнимость задач $\{\tau_i\}$. Хотя их выполнимость можно обеспечить, если назначать T_1 и T_2 не через посредничество допустимых СО, а напрямую через условие выполнимости, ведь выражение во второй строке условия (4) задает диапазон возможных значений T_i. Чем больше T_i, тем, как правило, проще обеспечить выполнимость, поэтому предлагается всякий раз максимизировать T_i. В рассматриваемом примере, учитывая $rx_1^{Lo} = rx_1^{Up} = 0$, получается, что $T_1 = 15$, $T_2 = 15$. Тогда $U_\tau^{Up} = 100\ \%$, что допустимо. При этом $xy_1^{Up} = 5$, $xy_2^{Up} = 15$, т. е. соблюдается неравенство в третьей строке условия (4) для обеих задач. Выбор значений $O_1 = 0$ и $O_2 = 0$ обеспечивает соблюдение выражения в первой строке условия (4) в случае обеих задач. Тем самым обеспечивается выполнимость задач $\{\tau_i\}$. Такой подход можно обобщить для любых ЛИО. Иногда может быть так, что указанный подход не обеспечивает выполнимость задач $\{\tau_i\}$, когда алгоритм А ее обеспечивает. В таких случаях можно просто воспользоваться алгоритмом А. Все это оформляется в виде алгоритма планирования, названного *алгоритмом АП*, так как в нем непосредственно используются ЛИО при анализе (А) выполнимости задач и формировании их периодов (П).

Алгоритм 4. Алгоритм АП, реализующий планирование при любых ЛИО.

1. Последовательно выполняются все действия из описания шага 1 алгоритма 1.

2. Приоритеты задачам назначаются по правилу: чем меньше D_i, тем выше приоритет.

3. Перебираются все задачи, имеющие ограничение Ξ_i, начиная с задачи с наиболее высоким приоритетом, и далее в порядке снижения приоритетов. В ходе этого перебора для очередной выбранной задачи τ_i, имеющей ограничение Ξ_i, выполняются следующие действия. С помощью алгоритма 2 формируется условие выполнимости и с помощью алгоритма 3 из работы [13], в составе входных данных которого $\beta = 0$, вычисляются те из оценок rx_i^{Lo}, rx_i^{Up}, ry_i^{Lo}, ry_i^{Up}, xy_i^{Lo} и xy_i^{Up}, которые встречаются в сформированном условии выполнимости. Эти оценки подставляются в данное условие. Затем отыскиваются такие O_i и T_i, что они удовлетворяют условию выполнимости для задачи τ_i, и значение T_i максимально возможное. Для этого решается соответствующая задача линейного программирования; если ее решения не существует, то выполняется переход к шагу 7.

4. Если $U_\tau^{Up} > 100\,\%$, то осуществляется переход к шагу 7.

5. Выполняется анализ выполнимости каждой задачи из множества $\{\tau_i\}$. Для этого в случае задачи τ_i, имеющей СО, проверяется условие $rf_i^{Up} \leqslant D_i$, а в случае задачи τ_i, имеющей ограничение Ξ_i, выполняются следующие действия. Прежде всего, с помощью алгоритма 3 из работы [13], в составе входных данных которого $\beta = 1$, вычисляются те из оценок rx_i^{Lo}, rx_i^{Up}, ry_i^{Lo}, ry_i^{Up}, xy_i^{Lo} и xy_i^{Up}, которые встречаются в условии выполнимости для задачи τ_i. Затем с учетом значений O_i и T_i и вычисленных оценок проверяется условие выполнимости для задачи τ_i.

6. Если все условия, проверенные на предыдущем шаге, были соблюдены, то выполнимость задач $\{\tau_i\}$ обеспечивается, и определяемый алгоритм завершается.

7. Выполняется алгоритм А применительно к задачам $\{\tau_i\}$. Определяемый алгоритм завершается.♦

7. АЛГОРИТМ АПП

Алгоритм АП реализует лишь один вариант назначения приоритетов (см. шаг 2 алгоритма 4), соответствующий принципу DM — Deadline Monotonic (см. обзор [1]), который оптимален при наличии только СО, когда $O_i = 0$, $D_i \leqslant T_i$. Этот принцип не оптимален при наличии ЛИО, хотя часто приводит к подходящему назначению приоритетов. Бывает, что алгоритм АП не может обеспечить выполнимость задач $\{\tau_i\}$, так как он пропускает другие варианты назначения приоритетов. Предлагается алгоритм планирования, названный *алгоритмом АПП* и основанный на алгоритме АП с выполнением полного (П) перебора вариантов назначения приоритетов.

Алгоритм 5. Алгоритм АПП, реализующий планирование при любых ЛИО.

1. Если еще не все варианты назначения приоритетов рассмотрены, то реализуется новый вариант назначения приоритетов, иначе выполняется переход к шагу 4.

2. Последовательно выполняются все действия из описания шагов 3—6 алгоритма 4 (АП) за исключением того, что переходы к шагу 7 заменяются переходами к шагу 1.

3. Выполняется переход к шагу 1.

4. Выполняется алгоритм А применительно к задачам $\{\tau_i\}$. Определяемый алгоритм завершается.♦

8. АЛГОРИТМ АПС

Выполнение алгоритма АПП можно ускорить, сделав перебор вариантов так, чтобы сначала проверялись варианты, имеющие больше шансов стать подходящими. Однако для n задач в наихудшем случае надо перебрать $n!$ вариантов, что при больших n практически нереализуемо. Предлагается алгоритм планирования, названный *алгоритмом АПС* и основанный на алгоритме АП с реализацией сокращенного (С) перебора вариантов назначения приоритетов.

Алгоритм 6. Алгоритм АПС, реализующий планирование при любых ЛИО.

1. Формируется начальный вариант назначения приоритетов. Сначала делается попытка использовать принцип назначения, реализуемый шагами 1 и 2 алгоритма 4 (АП), за исключением того, что если какое-то допустимое СО сформировать не удалось, то определяемый алгоритм не завершается, а применяется второй способ начального назначения приоритетов согласно правилу: чем меньше Csf_i^{Up}, тем выше приоритет. Устанавливается $w = 1$, где w — переменная, определяющая самый высокий приоритет, подлежащий обмену.

2. Последовательно выполняются все действия из описания шага 3 алгоритма 4 (АП) за исключением того, что переход к шагу 7 заменяется переходом к шагу 3.

3. Пусть задача называется *подготовленной*, если для нее сформированы значения O_i и T_i и обеспечивается ее выполнимость при анализе, осуществляемом согласно действиям шага 5 алгоритма 4 (АП). Если все задачи подготовленные, то переход к шагу 7.

4. Делается попытка подобрать другой вариант назначения приоритетов следующим образом (шаги 4, 5 и 6). Определяется задача τ_i, которая имеет

наивысший приоритет среди всех неподготовленных задач. Делается попытка назначить задаче τ_i более высокий приоритет, что, возможно, позволит ей стать подготовленной. Устанавливается $e = i$, где e — переменная, определяющая задачу τ_e, с которой возможен обмен приоритетами.

5. Если $\pi_e \leqslant w$, то осуществляется переход к шагу 8, так как не удается найти подходящий приоритет для неподготовленной задачи τ_i. Определяется индекс задачи с приоритетом $\pi_e - 1$. Переменная e устанавливается равной этому индексу. Выполняется обмен приоритетами между задачами τ_i и τ_e. Если задача τ_i имеет ЛИО, то выполняются действия из описания шага 3 алгоритма 4 (АП) применительно *только* к задаче τ_i, т. е. без перебора других задач, при этом также исключается возможный переход к другому шагу. Если теперь задача τ_i является подготовленной, то, сначала, устанавливается $w = \pi_i + 1$, а затем осуществляется переход к шагу 2.

6. Восстанавливается назначение приоритетов на основе повторного обмена приоритетами между задачами τ_i и τ_e. Выполняется переход к шагу 5.

7. Если $U_\tau^{Up} \leqslant 100$ %, то алгоритм завершается, обеспечив выполнимость задач $\{\tau_i\}$.

8. Выполняется алгоритм А применительно к задаче $\{\tau_i\}$. Определяемый алгоритм завершается. ♦

9. СРАВНЕНИЕ АЛГОРИТМОВ ПЛАНИРОВАНИЯ

Остается сравнить алгоритмы на основе имитационного моделирования. В ходе одного сеанса моделирования генерируется N случайных экземпляров множества задач $\{\tau_i\}$, и для каждого из них выполняются базовый алгоритм и алгоритмы А, АП, АПП и АПС. В итоге, определяется процент случаев, когда алгоритм обеспечивает выполнимость множества задач $\{\tau_i\}$, т. е. *успешность* рассматриваемого алгоритма, что можно считать оценкой эффективности алгоритма на совокупности генерируемых множеств задач $\{\tau_i\}$.

Были проведены сеансы моделирования с разными вариантами генерирования множества $\{\tau_i\}$ и разными вариантами настроек. Для примера рассмотрим следующий способ генерирования экземпляра множества $\{\tau_i\}$. В этом способе используются некоторые идеи из работы [14]. Индексы задачам даются в случайном порядке. Квант времени устанавливается равным 1. Пусть $\xi(\alpha, \beta)$ — целое случайное число с равномерным распределением на интервале $[\alpha, \beta]$. Пусть $\zeta = \min(1, \rho)$, где ρ — случайная величина, имеющая экспоненциальный закон распределения с параметром $\lambda = 10$. Пусть $T^{\min}$ и $T^{\max}$ — опорные параметры периодичности. Для каждой задачи τ_i устанавливаются значения $T^{\min} = 50$, $T^{\max} = 10\,000 \cdot 2^{i-1}$.

Параметры задачи τ_i, имеющей СО (1), формируются так: $T_i = \xi(T^{\min}, T^{\max})$, $O_i = \xi(0, T_i)$, $D_i = \xi(1, T_i)$, $Csf_i^{Up} = \xi(1, D_i)$, $Csf_i^{Lo} = \max(1, [(1 - \zeta) \cdot Csf_i^{Up}]_{Lo})$, где $[\alpha]_{Lo}$ — округление α до меньшего целого; остальные 10 оценок длительностей определяются очевидным образом, ведь для τ_i, имеющей СО (1), можно считать, что $x_{i,\nu} = s_{i,\nu}$, $y_{i,\nu} = f_{i,\nu}$ при $\forall \nu \geqslant 1$.

Пусть $C(\alpha, \beta)$ — процедура, генерирующая оценки длительностей компонентов запросов согласно правилу: $Csf_i^{Up} = \xi(\alpha, \beta)$; $Csf_i^{Lo} = \max(1, [(1 - \zeta) Csf_i^{Up}]_{Lo})$; $Csx_i^{Up} = [(Csf_i^{Lo} - 1)\zeta]_{Lo}$; $Cyf_i^{Up} = [(Csf_i^{Lo} - 1)\zeta]_{Lo}$; если $Csx_i^{Up} + Cyf_i^{Up} > Csf_i^{Lo}$, то $\gamma = (Csx_i^{Up} + Csf_i^{Lo} - Cyf_i^{Up})/2$, $Csx_i^{Up} = [\gamma - 1]^{Up}$, $Cyf_i^{Up} = Csf_i^{Lo} - [\gamma + 1]_{Lo}$, где $[\alpha]^{Up}$ обозначает округление α до большего целого; $Csx_i^{Lo} = [(1 - \zeta) \times Csx_i^{Up}]_{Lo}$; $Cyf_i^{Lo} = [(1 - \zeta) Cyf_i^{Up}]_{Lo}$; $Csy_i^{Up} = Csf_i^{Up} - Cyf_i^{Lo}$; $Cxf_i^{Up} = Csf_i^{Up} - Csx_i^{Lo}$; $Cxy_i^{Up} = Csf_i^{Up} - Csx_i^{Lo} - Cyf_i^{Lo}$; $Csy_i^{Lo} = Csf_i^{Lo} - Cyf_i^{Up}$; $Cxf_i^{Lo} = Csf_i^{Lo} - Csx_i^{Up}$; $Cxy_i^{Lo} = Csf_i^{Up} - Csx_i^{Lo} - Cyf_i^{Up}$.

Для задач, имеющих ЛИО, генерируются его параметры и оценки длительностей. Например, в случае задачи τ_i, имеющей НО (3), $T_i^{xx\max} = \xi(T^{\min}, T^{\max})$, $T_i^{xx\min} = \xi(0, T_i^{xx\max})$, $T_i^{xy\max} = \xi(T^{\min}, T^{\max})$, $x_{i,0} = 0$, и вызывается процедура $C(1, \min(T_i^{xx\max}, T_i^{xy\max}))$.

Пусть U_τ^* обозначает *опорную* загрузку. Для каждой задачи τ_i, имеющей ЛИО, формируется допустимое СО и устанавливается $T_i = T_i^*$. Затем вычисляется U_τ^{Up} согласно формуле (5). Если $|U_\tau^{Up} - U_\tau^*| \leqslant 1$ %, то экземпляр множества $\{\tau_i\}$ сгенерирован, иначе делается попытка выполнить указанное неравенство путем пропорционального уменьшения 12-ти оценок длительностей компонентов запросов всех задач из множества $\{\tau_i\}$. Если не удается добиться выполнения этого неравенства, то тогда генерирование экземпляра множества $\{\tau_i\}$ начинается с самого начала.

На рис. 4 приведены результаты трех экспериментов. В эксперименте 1 каждый экземпляр мно-

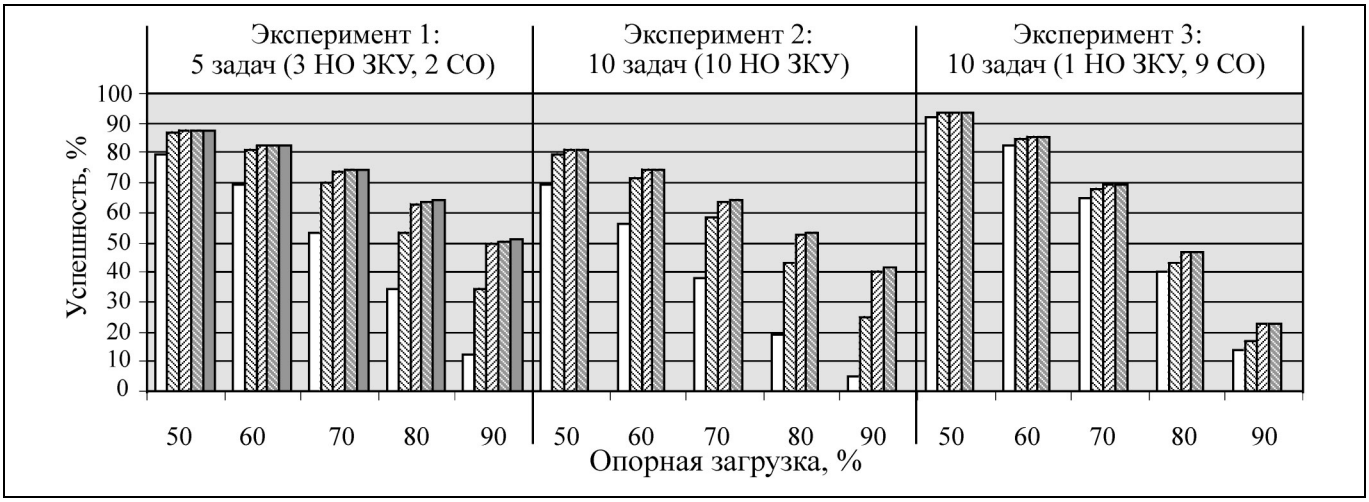

Рис. 4. Пример результатов имитационного моделирования:
□ — базовый алгоритм; ▨ — алгоритм А; ▨ — алгоритм АП; ▨ — алгоритм АПС; ▨ — алгоритм АПП

жества $\{\tau_i\}$ состоял из 3-х задач, имеющих НО (3), и двух задач, имеющих СО (1). В эксперименте 2 — из 10-ти задач, имеющих НО (3). В эксперименте 3 — из 1-й задачи, имеющей НО (3), и 9-ти задач, имеющих СО (1). В каждом эксперименте проводились 5 сеансов моделирования для опорной загрузки U_τ^* от 50 до 90 % с шагом 10 %. В случае каждого сеанса устанавливалось $\theta = 1$ и генерировалось $N = 10\,000$ случайных экземпляров множества $\{\tau_i\}$ согласно рассмотренному способу. В эксперименте 1 преимущество алгоритма АПС по сравнению с базовым составляет от 8 до 38 %. Это означает, что при $U_\tau^* = 90$ % алгоритм АПС в 38 % случаев обеспечивает выполнимость множества задач $\{\tau_i\}$, когда базовый алгоритм это сделать не может, т. е. применение алгоритма АПС вместо базового позволяет во многих случаях избежать затрат на приобретение более быстродействующего вычислительного устройства. В этом проявляется существенное повышение эффективности планирования при алгоритме АПС. В экспериментах 2 и 3 в каждом множестве $\{\tau_i\}$ 10 задач, что сильно замедляет алгоритм АПП и поэтому не позволяет его применять. Эксперимент 2 отражает ситуацию, когда все задачи имеют ЛИО. Видно, что его результаты во многом подобны результатам эксперимента 1, где в каждом множестве $\{\tau_i\}$ только 60 % задач имеют ЛИО. Важно, что разработанные алгоритмы обеспечивают заметное повышение эффективности планирования, даже когда среди 10-ти задач только одна задача имеет ЛИО (эксперимент 3). Здесь преимущество достигает 8—9 % в случае алгоритмов АП, АПС. Схожие результаты получены в экспериментах при других настройках моделирования, которые, в частности, отличаются видами ЛИО, числом задач, способом реализации процедуры $C(\alpha, \beta)$.

Разница между успешностью алгоритмов АП, АПС и АПП не столь велика: алгоритмы АПС и АПП имеют преимущество над АП до 0,8 и 1,2 % соответственно. Однако трудно найти причину не применять алгоритм АПС вместо АП на этапе проектирования, ведь они оба имеют приемлемый уровень быстродействия, и есть вероятность, что алгоритм АПС успешно сработает там, где алгоритм АП не справится. При небольшом числе задач рекомендуется применять алгоритм АПП, имеющий наилучшую успешность. Алгоритмы А и АП также имеют практическое значение, в частности, для адаптивных систем РВ, где требуется выполнять алгоритмы планирования в ходе работы системы, а не на этапе проектирования. Поэтому могут понадобиться более простые и быстрые алгоритмы, пусть и с несколько меньшими показателями эффективности.

ЗАКЛЮЧЕНИЕ

Выделен класс линейных интервальных ограничений, который охватывает многие виды нестандартных ограничений, встречающиеся на практике, и не входит в целевые классы нестандартных ограничений известных подходов. Разработан ряд алгоритмов, обеспечивающих планирование с фиксированными приоритетами при любых линейных интервальных ограничениях. На основе имитационного моделирования показано, что разработанные алгоритмы обеспечивают существенное повышение эффективности планирования по сравнению с базовым алгоритмом. Предложенные алгоритмы характеризуются разными соотношениями «эффективность—быстродейст-

вие», поэтому в условиях планирования с фиксированными периодическими приоритетами при линейных интервальных ограничениях теперь есть хорошая возможность выбора подходящего алгоритма планирования.

ЛИТЕРАТУРА

1. *Real-Time* Scheduling Theory: A Historical Perspective / L. Sha, T. Abdelzaher, K.E. Årzén et al. // Real-Time Systems. — 2004. — N 28. — P. 101—155.
2. *Dobrin R., Fohler G., Puschner P.* Translating Off-line Schedules into Task Attributes for Fixed Priority Scheduling // Proc. of 22nd IEEE Real-Time Systems Symposium. — London, 2001. — P. 225—234.
3. *Wang W., Mok A.K., Fohler G.* Generalized Pre-Scheduler // Proc. of The 16th Euromicro Conference on Real-Time Systems. — Catania, 2004. — P. 127—134.
4. *Sandström K., Norström C.* Managing Complex Temporal Requirements in Real-Time Control Systems // Proc. of The 9th IEEE Conference on Engineering of Computer-Based Systems. — Lund, 2002. — P. 103—109.
5. *Kavalerov M., Matushkin N.* A Formal Representation of Standard Timing Constraints for Periodic Real-Time Tasks // Acta Universitatis Pontica Euxinus. — 2006. — Vol. VI, N 7. — P. 20—22.
6. *Gerber R., Hong S.* Semantics-Based Compiler Transformations for Enhanced Schedulability // Proc. of The 14th IEEE Real-Time Systems Symposium. — San Juan, 1993. — P. 232—242.
7. *Кавалеров М.В., Матушкин Н.Н.* Применение обобщенных нестандартных ограничений реального времени в условиях планирования с фиксированными приоритетами // Информационные технологии моделирования и управления. — 2005. — № 6 (24). — С. 842—848.
8. *Jitter* Compensation for Real-Time Control Systems / P. Marti, G. Fohler, K. Ramamritham, J.M. Fuertes // Proc. of The 22nd IEEE Real-Time Systems Symposium. — London, 2001. — P. 39—48.
9. *Fohler G.* Dynamic Timing Constraints — Relaxing Over-constraining Specifications of Real-Time Systems // Proc. of Work-in-Progress Session, The 18th IEEE Real-Time Systems Symposium. — San Francisco, 1997. — P. 27—30.
10. *Bate I., Burns A.* An Approach to Task Attriβute Assignment for Uniprocessor Systems // Proc. of The 11th Euromicro Conference on Real-Time Systems. — York, 1999. — P. 46—53.
11. *Кавалеров М.В.* Преобразование линейных интервальных ограничений реального времени в стандартные ограничения // Системы управления и информационные технологии. — 2006. — № 4.2 (26). — С. 228—233.
12. *Audsley N.C.* Optimal Priority Assignment and Feasibility of Static Priority Tasks with Arbitrary Start Times. Technical Report YCS164, Department of Computer Science, University of York, UK. — 1991. — 31 p.
13. *Кавалеров М.В., Матушкин Н.Н.* Вычисление оценок параметров выполнения запросов в условиях планирования с фиксированными приоритетами // Системы мониторинга и управления: Сб. науч. тр. Перм. гос. техн. ун-та. — Пермь, 2006. — С. 154—164.
14. *Davis R.I.* Approximate Slack Stealing Algorithms for Fixed Priority Pre-emptive Systems, Technical Report YCS216, Department of Computer Science, University of York, UK. — 1993. — 27 p.

e-mail: mkavalerov@gmail.com

Статья представлена к публикации членом редколлегии В.Д. Малюгиным. ❑