ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2023, № 1, с. 72–81

СИСТЕМНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

УДК 519.86

ОРГАНИЗАЦИЯ ПОДСИСТЕМЫ КОНТРОЛЯ В ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

© 2023 г. М. Г. Фуругян

Федеральный исследовательский центр "Информатика и управление" РАН, Москва, Россия

e-mail:rtsccas@yandex.ru

Поступила в редакцию 27.06.2022 г.
После доработки 12.08.2022 г.
Принята к публикации 26.09.2022 г.

Рассматривается задача оптимального расположения модулей контроля в вычислительной системе реального времени. Данная задача формулируется в виде минимаксной. Исследуются различные структуры графа частичного порядка выполнения прикладных модулей: последовательная цепочка, несколько независимых последовательных цепочек, дерево, ориентированное от корня к листьям, дерево, ориентированное от листьев к корню, произвольный граф без циклов. Разработаны алгоритмы построения оптимальной расстановки модулей контроля.

DOI: 10.31857/S000233882301002X, **EDN:** IZTYEW

Введение. В настоящее время вычислительные системы реального времени применяются в различных областях деятельности человека: в гражданском и военном строительстве, при испытаниях и эксплуатации самолетов и ракет, при проектировании и функционировании производственных и транспортных систем, в нефте- и газодобывающих отраслях. Одно из основных требований, предъявляемых к таким системам, заключается в том, что все вычисления должны производиться при заданных временных ограничениях. Для решения этой задачи могут быть использованы различные методы построения оптимальных расписаний [1–4].

В [5–7] разработана методика проверки выполнения ограничений реального времени, заключающихся в том, что каждая работа должна выполняться в заданном директивном интервале. Проведенные исследования выполнены для многоядерной вычислительной системы реального времени и основаны на построении имитационной модели с использованием обобщенных конечных автоматов с остановкой таймера. С помощью этой модели строится временная диаграмма работы системы, позволяющая осуществить непосредственную проверку выполнения ограничений реального времени. В [8] предложен псевдополиномиальный алгоритм решения задачи построения оптимального по быстродействию расписания исполнения заданий с логическими условиями предшествования. В этой задаче для каждого задания дан список его непосредственных предшественников, а также число завершенных непосредственных предшественников, необходимое для начала его выполнения. Задача сведена к циклической игре. В [9, 10] некоторые задачи планирования работ сведены к минимаксным задачам.

Помимо временных ограничений на выполняемые вычисления, при создании систем реального времени необходимо учитывать возможность возникновения различных сбоев и проводить проверку правильности работы прикладных модулей. Для этого в систему кроме прикладных модулей вводят модули контроля – специальные программы, отслеживающие сбои и ошибки и сохраняющие промежуточную информацию. Так, например, при проведении летных испытаний некоторые модули контроля выполняют проверку того, что вычисляемые параметры принадлежат области допустимых значений и удовлетворяют различным соотношениям. Это позволяет в случае необходимости рестарта системы не выполнять все вычисления заново, а использовать данные, сохраненные модулями контроля. Таким образом, наличие модулей контроля может существенно улучшить надежность работы системы и уменьшить временные издержки, вызванные возникновением сбоев и ошибок, что крайне важно для систем жесткого реального времени.

При разработке вычислительных систем реального времени возникает задача оптимизации расположения модулей контроля среди прикладных модулей. Эта задача исследовалась ранее ря-

дом авторов с использованием вероятностных методов. При этом предполагается, что известны вероятности возникновения сбоев и ошибок. В [11, 12] допускалось, что вероятность возникновения ошибки в одном и том же месте системы не является постоянной и может изменяться после рестарта. В [13] для многопроцессорной системы определяется такое расположение модулей контроля, при котором вероятность выполнить все прикладные модули до выхода из строя всех процессоров максимальна. В [14] определяется понятие зоны рестарта – совокупность прикладных модулей, которые следует выполнить заново в случае обнаружения ошибки некоторым модулем контроля. В предположении, что известно расположение модулей контроля, строится минимальная зона рестарта. В [15] решается задача расположения модулей контроля с целью минимизировать математическое ожидание суммарного времени, затраченного на выполнение всех модулей (включая их повторное выполнение при рестарте). Рассмотрены различные виды графа частичного порядка выполнения прикладных модулей.

В настоящей статье рассматривается вычислительная система реального времени, состоящая из прикладных модулей, на множестве которых задано отношение частичного порядка в виде ориентированного графа без циклов. Предполагается, что число модулей контроля фиксировано, и временем их выполнения можно пренебречь, поскольку оно существенно меньше времени выполнения прикладных модулей. Каждый модуль контроля располагается непосредственно после некоторого прикладного модуля. При обнаружении ошибки каким-либо модулем контроля определяется область повторного выполнения прикладных модулей. Эта область состоит из одного или нескольких путей графа, ведущих в модуль контроля, обнаруживший ошибку. Требуется таким образом расположить модули контроля, чтобы максимально возможная длина такого пути была минимальной. Данная задача формулируется в виде минимаксной задачи. Исследуются различные структуры графа частичного порядка выполнения прикладных модулей. Разработаны алгоритмы построения оптимальной расстановки модулей контроля.

1. Постановка задачи. Задан набор прикладных модулей $W = \{w_1, w_2, \cdots, w_N\}$, $N \geq 2$, подлежащих выполнению на вычислительной системе. Число процессоров в системе может быть произвольным. (Как будет показано ниже, на решение рассматриваемой задачи это не влияет.) Каждый модуль w_i выполняется без прерываний и переключений с одного процессора на другой, а длительность его выполнения равна $t_i = t(w_i)$, $i = \overline{1, N}$. На множестве W задано отношение частичного порядка выполнения прикладных модулей в виде ориентированного графа без циклов $G = (W, A)$, где W – множество узлов, A – множество ориентированных дуг. Если $(w_i, w_j) \in A$, то начало исполнения модуля w_j возможно только после завершения модуля w_i. В этом случае будем также использовать обозначение $w_i \to w_j$. В цепочке

$$w_{i_1} \to w_{i_2} \to \ldots \to w_{i_{n-1}} \to w_{i_n} \tag{1.1}$$

модули w_{i_j} при $j < p$, $j = \overline{1, n-1}$, являются предшественниками модуля w_{i_p}, а при $j > p$, $j = \overline{2, n}$, – его последователями. Кроме того, модуль w_{i_j} выступает непосредственным предшественником $w_{i_{j+1}}$, а $w_{i_{j+1}}$ – непосредственным последователем w_{i_j}, $j = \overline{1, n-1}$. Введем обозначения:

$$P(w_i) = \{w_j \in W : w_j \to w_i\}, \quad Q(w_i) = \{w_j \in W : w_i \to w_j\},$$

$$P_0 = \{w_i \in W : P(w_i) = \varnothing\}, \quad Q_0 = \{w_i \in W : Q(w_i) = \varnothing\},$$

т.е. $P(w_i)$ и $Q(w_i)$ – соответственно все непосредственные предшественники и непосредственные последователи прикладного модуля w_i, а P_0 и Q_0 – прикладные модули, не имеющие соответственно непосредственных предшественников и непосредственных последователей.

Помимо прикладных модулей в системе имеется K модулей контроля, $K \leq N$. Каждый модуль контроля присоединяется к некоторому прикладному модулю и выполняется сразу же после завершения этого прикладного модуля. К каждому прикладному модулю присоединяется не более одного модуля контроля. Предполагается, что временем выполнения модуля контроля можно пренебречь, поскольку оно существенно меньше времени выполнения прикладных модулей. Пусть $\overline{W} = \{w_{i_1}, w_{i_2}, \ldots, w_{i_K}\}$ – множество всех прикладных модулей, к которым присоединены модули контроля. Будем называть его расстановкой модулей контроля или просто расстановкой.

Модуль контроля после своего выполнения сигнализирует об отсутствии или наличии ошибки. В первом случае вычисления продолжаются по ранее построенному расписанию. В случае

обнаружения модулем контроля ошибки необходимо повторить (инициировать рестарт) выполнение некоторых прикладных модулей следующим образом.

Предположим, что модулем контроля, присоединенным к некоторому прикладному модулю $w_{i_n} \in \overline{W}$, была обнаружена ошибка. В графе G рассматриваются все ориентированные пути (цепочки) π вида (1.1), в которых $w_{i_j} \notin \overline{W}$, $j = \overline{1, n-1}$. Кроме того, либо $w_{i_1} \in P_0$, либо модуль w_{i_1} имеет одного или нескольких предшественников, и если $w_{i_1-1} \to w_{i_1}$, то $w_{i_1-1} \in \overline{W}$. Иными словами, указанный путь начинается с прикладного модуля, который либо не имеет непосредственных предшественников, либо к каждому из его непосредственных предшественников присоединен модуль контроля, а все остальные прикладные модули этого пути, кроме w_{i_n} ($w_{i_n} \in \overline{W}$), не являются таковыми. Будем называть такие пути π путями (или цепочками) рестарта. В случае обнаружения ошибки модулем контроля, присоединенным к прикладному модулю w_{i_n}, необходимо для каждой цепочки рестарта вида (1.1) повторить выполнение всех ее прикладных модулей. Длиной цепочки рестарта π вида (1.1) будем называть величину

$$t(\pi) = \sum_{j=1}^{n} t_{i_j}. \tag{1.2}$$

Пусть $\Pi(w_{i_n})$ – множество всех путей рестарта π вида (1.1) и пусть

$$t(\Pi(w_{i_n})) = \max_{\pi \in \Pi(w_{i_n})} t(\pi),$$

т.е. $t(\Pi(w_{i_n}))$ – это длина максимального из указанных путей. Допустимой расстановкой будем называть такую расстановку $\overline{W}$, при которой к каждому прикладному модулю из Q_0 присоединен модуль контроля, т.е.

$$Q_0 \subseteq \overline{W}. \tag{1.3}$$

Отметим, что при $K < |Q_0|$ допустимой расстановки не существует. Если расстановка не является допустимой, то контроль вычислений не может быть выполнен полностью. Поэтому всюду в дальнейшем будем предполагать, что $K \geq |Q_0|$ и что к каждому прикладному модулю из Q_0 присоединен модуль контроля.

Пусть Ω – множество всех допустимых расстановок. Задача заключается в выборе допустимой расстановки $\overline{W}$, для которой величина

$$\max_{w \in W} t(\Pi(w))$$

минимальна. Иными словами, требуется определить величину

$$T_{opt} = \min_{W \in \Omega} \max_{w \in W} t(\Pi(w)) \tag{1.4}$$

и допустимую расстановку $\overline{W}$, на которой реализуется указанный минимакс. Такую расстановку будем называть оптимальной. Таким образом, при оптимальной расстановке минимизируется максимальная длина (1.2) цепочки рестарта.

Замечание 1. В данной постановке задачи при любом числе процессоров время выполнения рестарта в случае обнаружения ошибки не может быть меньше длины максимальной цепочки рестарта. Поэтому представленная минимаксная постановка актуальна независимо от числа процессоров в системе.

Замечание 2. В данной постановке задачи предполагается, что в случае обнаружения ошибки необходимо повторно выполнить прикладные модули, принадлежащие описанным выше цепочкам рестарта, независимо от того, какой результат выдали другие модули контроля.

2. Построение оптимальной расстановки. Предлагаются методы построения оптимальной расстановки для случаев, когда граф G представляет собой последовательную цепочку прикладных модулей, несколько независимых последовательных цепочек, дерево, ориентированное от листьев к корню, дерево, ориентированное от корня к листьям, и произвольный граф без циклов.

2.1. Последовательная цепочка. Допустим, что граф G является последовательной цепочкой $w_1 \to w_2 \to \ldots \to w_N$. В этом случае $Q_0 = \{w_N\}$ и в силу (1.3) любая допустимая расста-

новка имеет вид $\overline{W} = \{w_{i_1}, w_{i_2}, \ldots, w_{i_{K-1}}, w_{i_K} = w_N\}$. Без ограничения общности можно считать, что $K \geq 2$ и тогда $\Omega = \{\overline{W} : 1 \leq i_1 < i_2 < \cdots < i_{K-1} < i_K = N\}$. Введем обозначения:

$$I_k = \{i_{k-1} + 1, \ldots, i_k\}, \quad T_k = \sum_{i \in I_k} t_i, \tag{2.1}$$

где $k = \overline{1, K}$, $i_0 = 0$. В соответствие с (1.4) задача заключается в поиске величины

$$T_{opt} = \min_{W \in \Omega} \max_{k=\overline{1,K}} T_k \tag{2.2}$$

и расстановки $\overline{W}$, реализующей указанный минимакс. Ниже предлагаются два алгоритма решения этой задачи – точный (алгоритм 1) и приближенный (алгоритм 2).

Алгоритм 1 основан на переборе всех допустимых расстановок, число которых составляет C_{N-1}^{K-1}. Алгоритм перебора базируется на том, что величины $i_1, i_2, \ldots, i_K$ могут принимать следующие значения: $i_K = N$, $i_{K-1} = \overline{K-1, i_K - 1}$, $i_{K-2} = \overline{K-2, i_{K-1} - 1}$, ..., $i_1 = \overline{1, i_2 - 1}$. Для каждого набора $i_1, i_2, \ldots, i_K$ вычисляются величины T_k, $k = \overline{1, K}$, согласно (2.1), и величина T_{opt}, согласно (2.2). Расстановка $\overline{W}$, реализующая минимакс (2.2), будет оптимальной.

Определим вычислительную сложность алгоритма 1. Для каждой расстановки при вычислении величин T_k, $k = \overline{1, K}$, требуется выполнить $O(N)$ операций, а для вычисления максимальной из величин T_k – $O(K)$ операций. Таким образом, вычислительная сложность алгоритма 1 составляет $O(C_{N-1}^{K-1}(N + K))$ или $O((N - K + 1)\cdots(N - 1)(N + K)/(K - 1)!)$. Поскольку $K \leq N$, то при фиксированном K сложность алгоритма 1 составляет $O(N^K)$, т.е. алгоритм 1 имеет полиномиальную вычислительную сложность.

Далее предлагается приближенный алгоритм (алгоритм 2). Введем обозначения:

$$T_1^* = \left(\sum_{i=1}^{N} t_i\right)/K, \tag{2.3}$$

$$T_2^* = \max_{i=\overline{1,N}} t_i; \quad T^* = \max(T_1^*, T_2^*).$$

Каждая из величин T_1^*, T_2^* и T^* является нижней оценкой величины T_{opt}, т.е.

$$T_{opt} \geq T^*. \tag{2.4}$$

Действительно, если $T_{opt} < T_1^*$, то существует допустимая расстановка $\overline{W} \in \Omega$, для которой $T_k < T_1^*$ при всех $k = \overline{1, K}$ и, следовательно,

$$\sum_{k=1}^{K} T_k < KT_1^*, \quad \sum_{i=1}^{N} t_i < KT_1^*.$$

Откуда следует, что

$$\left(\sum_{i=1}^{N} t_i\right)/K < T_1^*,$$

что противоречит (2.3). Следовательно,

$$T_{opt} \geq T_1^*. \tag{2.5}$$

Кроме того, для любой допустимой расстановки $\overline{W}$ при некотором k, $1 \leq k \leq K$, справедливо неравенство $T_2^* \leq T_k$. Следовательно,

$$T_{opt} \geq T_2^*. \tag{2.6}$$

Из (2.5), (2.6) следует (2.4).

Алгоритм 2 минимизирует максимальное отклонение величин T_k, $k = \overline{1, K}$, от T^*, т.е. находит

$$\min_{W \in \Omega} \max_{k=\overline{1,K}} |T_k - T^*|$$

и расстановку $\overline{W}$, реализующую этот минимакс.

Алгоритм 2.

Шаг 1. Положить $k = K$, $i_k = N$.

Шаг 2. Если $t_{i_k} \geq T^*$, то положить $I_k = \{i_k\}$, $l = i_k - 1$, перейти на шаг 5. В противном случае (т.е. если $t_{i_k} < T^*$) перейти на шаг 3.

Шаг 3. Определить номер i_0, $1 < i_0 \leq i_k$, для которого

$$\sum_{i=i_0}^{i_k} t_i \leq T^*, \quad \sum_{i=i_0-1}^{i_k} t_i \geq T^*.$$

Если такой номер существует, то перейти на шаг 4, в противном случае – на шаг 6.

Шаг 4. Положить

$$\Delta_1 = T^* - \sum_{i=i_0}^{i_k} t_i, \quad \Delta_2 = \sum_{i=i_0-1}^{i_k} t_i - T^*.$$

Если $\Delta_1 \leq \Delta_2$, то положить $l = i_0$, $I_k = \{i_0, \cdots, i_k\}$. Если $\Delta_1 > \Delta_2$, то положить $l = i_0 - 1$, $I_k = \{i_0 - 1, \cdots, i_k\}$.

Шаг 5. Если $l > 1$ и $k > 1$, то положить $k = k - 1$, $i_k = l$, перейти на шаг 2. В противном случае (т.е. если $l = 1$ или $k = 1$) перейти на шаг 6.

Шаг 6. Положить

$$I_1 = \{1, 2, \cdots, N\} \backslash \bigcup_{k=2}^{K} I_k.$$

Шаг 7. Если $k \geq 2$, то $(K - k)$ неиспользованных модулей контроля включить в максимальные по длине цепочки рестарта с помощью процедуры включения модуля контроля в цепочку рестарта, описание которой содержится в разд. 2.2.

Шаг 8. Решением задачи является допустимая расстановка $\overline{W} = \{w_{i_1}, w_{i_2}, \ldots, w_{i_N}\}$. Алгоритм завершен.

Поскольку для каждого k выполняется $O(N)$ операций, то сложность алгоритма 2 составляет $O(KN)$. Следовательно, алгоритм 2 имеет полиномиальную вычислительную сложность. Для определения отклонения величины

$$T = \max_{k=1,\,K} T_k$$

от T_{opt} применима оценка

$$T - T_{opt} \leq T - T^*, \tag{2.7}$$

которая может быть вычислена после работы алгоритма 2.

2.2. Несколько последовательных цепочек. Предполагается, что граф G состоит из p независимых (непересекающихся) цепочек:

$$w_{i1} \to w_{i2} \to \cdots \to w_{iN_i}, \quad i = \overline{1,\ p}. \tag{2.8}$$

В этом случае $P_0 = \{w_{i1}, i = \overline{1, p}\}$, $Q_0 = \{w_{iN_i}, i = \overline{1, p}\}$, а в силу (1.3) $\{w_{iN_i}, i = \overline{1, p}\} \subseteq \overline{W}$ и $K \geq p$. В приводимом ниже описании приближенного алгоритма построения оптимальной расстановки используется следующая процедура.

Процедура включения модуля контроля в цепочку рестарта. Пусть в цепочку рестарта (1.1) требуется включить дополнительный модуль контроля. Согласно (1.4), этот модуль контроля должен разбивать указанную цепочку на две цепочки так, чтобы длина максимальной из них была бы минимально возможной. Для этого вычисляются величины

$$\tau^* = \left(\sum_{j=1}^{n} t(w_{i_j})\right)/2, \quad S(r) = \sum_{j=1}^{r} t(w_{i_j}), \quad \Delta(r) = \left|\tau^* - S(r)\right|, \quad 1 \leq r \leq n.$$

Далее определяется номер r_0, $1 \leq r_0 \leq n$, при котором величина $\Delta(r)$ принимает минимальное значение, т.е.

$$\Delta(r_0) = \min_{r=1,n} \Delta(r).$$

Дополнительный модуль контроля присоединяется к прикладному модулю $w_{i_{n_0}}$. Процедура завершена.

Перейдем к описанию алгоритма построения оптимальной расстановки для случая нескольких независимых цепочек.

Алгоритм 3.

Шаг 1. Присоединить модули контроля к прикладным модулям w_{iN_i}, $i = \overline{1, p}$.

Шаг 2. Положить $l = K - p$. Если $l = 0$, алгоритм завершен. Если $l > 0$, то l раз повторять шаг 3.

Шаг 3. Среди всех цепочек рестарта вида (1.1), содержащих более одного прикладного модуля ($n > 1$), выбрать цепочку максимальной длины и выполнить для нее процедуру включения модуля контроля.

Алгоритм завершен.

Для определения вычислительной сложности алгоритма 3 отметим, что сложность шага 1 составляет $O(p)$, сложность вычисления длин цепочек рестарта и нахождения максимальной из них составляет

$$O\left(\sum_{i=1}^{p} N_i\right),$$

а сложность процедуры включения модуля контроля составляет

$$O(\max_{i=1,\ p} N_i)$$

и повторяется она $K - p$ раз. Следовательно, сложность алгоритма 3 составляет

$$O\left(\left(\sum_{i=1}^{p} N_i\right)(K - p)\right),$$

т.е. алгоритм 3 имеет полиномиальную вычислительную сложность.

Для оценки точности алгоритма 3 может быть использовано неравенство (2.7), где T – длина максимальной цепочки рестарта, полученной после работы алгоритма, а

$$T^* = \max\left[\left(\sum_{i=1}^{p}\sum_{j=1}^{N_i} t(w_{ij})\right)/K,\ \max_{i=1,\ p;\ j=1,\ N_i} t(w_{ij})\right].$$

Аналогичный подход применим к П-сетям, т.е. к сетям, в которых помимо цепочек (2.8) есть еще два узла w_0 и w_N, связанные с узлами w_{i1} и w_{iN_i} отношениями предшествования $w_0 \to w_{i1}$ и $w_{iN_i} \to w_N$, $i = \overline{1, p}$. В этом случае $P_0 = \{w_0\}$, $Q_0 = \{w_N\}$, $K \geq 1$.

Приведем еще один приближенный алгоритм, основанный на использовании алгоритма 2.

Алгоритм 4.

Шаг 1. Присоединить модули контроля к прикладным модулям w_{iN_i}, $i = \overline{1, p}$.

Шаг 2. Вычислить длины T_i, $i = \overline{1, p}$, всех цепочек вида (2.8).

Шаг 3. Каждой цепочке вида (2.8) дополнительно выделить k_i, $i = \overline{1, p}$, модулей контроля, где k_i вычисляется по следующим правилам:

а) $k_i \leq N_i - 1$;

б) величины k_i пропорциональны величинам T_i;

в) $\sum_{i=1}^{p} k_i = K - p$.

Шаг 4. В каждую цепочку вида (2.8) дополнительно включить k_i, $i = \overline{1, p}$, модулей контроля, применяя алгоритм 2. Алгоритм завершен.

Сложность алгоритма 4 составляет

$$O\left(\sum_{i=1}^{p} k_i N_i\right),$$

или

$$O\left(\left(\sum_{i=1}^{p} N_i\right)(K-p)\right),$$

т.е. алгоритм 4 имеет полиномиальную вычислительную сложность.

Оценка точности вычисляется с помощью неравенства (2.7) при

$$T^* = \max_{i=1,\ p}\left(T_i/k_i, \max_{j=1,\ N_i} t(w_{ij})\right).$$

2.3. Дерево, ориентированное от листьев к корню. Граф G имеет структуру дерева, ребра которого ориентированы от листьев к корню w_N. В данном случае $Q_0 = \{w_N\}$, P_0 – это множество всех листьев дерева. Каждый узел $w \in W$ дерева имеет два параметра: длительность $t(w)$ выполнения прикладного модуля w (заданная величина) и расстояние $s(w)$ (длина ориентированного пути) от w до ближайшего модуля контроля (вычисляется в ходе работы алгоритма). Приведем общее описание приближенного алгоритма построения оптимальной расстановки, а затем более подробно остановимся на каждом шаге алгоритма.

Алгоритм 5.

Шаг 1. Присоединить модуль контроля к корню дерева w_N.

Шаг 2. Для каждого узла $w \in W$ вычислить $s(w)$.

Шаг 3. Выполнять шаги 3.1, 3.2 $(K-1)$ раз.

Шаг 3.1. Найти максимальную цепочку рестарта, состоящую более чем из одного узла, и выполнить процедуру включения в нее модуля контроля. Пусть модуль контроля был присоединен к прикладному модулю $w_{i_0} \in W$.

Шаг 3.2. Для каждого узла w цепочек рестарта, ведущих к узлу w_{i_0}, пересчитать величину $s(w)$.

Алгоритм завершен.

В результате работы алгоритма 5 будет построена расстановка $\overline{W}$ и вычислена величина

$$T = \max_{w \in W} s(w).$$

Приведем пояснения к шагам алгоритма 5.

1. Поскольку $Q_0 = \{w_N\}$, то модуль контроля присоединяется к прикладному модулю w_N.

2. Величины $s(w)$ для $w \in W$ вычисляются по следующим правилам: $s(w_N) = t(w_N)$; если $\overline{w} \in P(w)$, то положить $s(\overline{w}) = s(w) + t(\overline{w})$.

3. После выполнения шага 2 величина $s(w)$ равна расстоянию от w до ближайшего модуля контроля.

3.1. Длина максимальной цепочки рестарта – это максимум величин $s(w)$ по всем $w \in W$, для которых либо $w \in P_0$, либо $\overline{w} \in \overline{W}$ для всех $\overline{w} \in P(w)$. Сама цепочка рестарта однозначно строится, следуя по ребрам дерева от w до ближайшего модуля контроля.

3.2. а) Положить $s(w) = s(w) - s(w_{i_0}) + t(w_{i_0})$ для каждого узла w, $w \neq w_{i_0}$, цепочек рестарта, ведущих к узлу w_{i_0}.

б) Положить $s(w_{i_0}) = t(w_{i_0})$.

Сложность алгоритма 5 составляет $O(NK)$, т.е. алгоритм имеет полиномиальную вычислительную сложность. Точность алгоритма 5 определяется с помощью оценки (2.7), где

$$T^* = \max\left(\left(\sum_{i=1}^{N} t(w_i)\right)/K, \max_{i=1,\ N} t(w_i)\right). \tag{2.9}$$

2.4. Дерево, ориентированное от корня к листьям. Граф G имеет структуру дерева, ребра которого ориентированы от корня w_N к листьям V, $V \subseteq W$. В этом случае $P_0 = \{w_N\}$, $Q_0 = V$, $K \geq |V|$. Поэтому к каждому листу присоединяется модуль контроля, т.е. $V \subseteq \overline{W}$. Каждый узел $w \in W$ имеет три параметра: $t(w)$, $\sigma(w)$ и $w'(w)$. Как и прежде, $t(w)$ – длительность выполнения модуля w. Значения параметров $\sigma(w)$ и $w'(w)$ определяются следующим образом. Если

$w \notin \overline{W}$, то $\sigma(w)$ – это длина максимальной цепочки $w \to w_{i_1} \to \cdots \to w_{i_n} \to v$ при $n > 0$, где $v \in \overline{W}$, а $w_{i_j} \notin \overline{W}$, $j = \overline{1, n}$, т.е. $\sigma(w) = t(w) + t(w_{i_1}) + \ldots + t(w_{i_n}) + t(v)$, и $w'(w) = w_{i_1}$. При $n = 0$ указанная цепочка имеет вид $w \to v$, и в этом случае $\sigma(w) = t(w) + t(v)$, $w'(w) = v$. Если $w \in \overline{W}$, то $\sigma(w) = t(w)$ и $w'(w) = w$. Значения параметров $\sigma(w)$ и $w'(w)$ вычисляются в ходе работы алгоритма.

Перед тем, как дать общее описание приближенного алгоритма построения оптимальной расстановки, остановимся на некоторых его частях. Как отмечено выше, все листья V дерева G включаются в $\overline{W}$, поскольку они не имеют непосредственных последователей. Вершины W дерева G разбиваются на непересекающиеся подмножества (уровни) $W_1, W_2, \cdots, W_M$, $W_i \cap W_j = \varnothing$ при $i \neq j$:

$$\bigcup_{i=1}^{M} W_i = W$$

следующим образом: $W_1 = \{w_N\}$,

$$W_m = \bigcup_{w \in W_{m-1}} Q(w), \quad m = \overline{2, M}$$

(т.е. каждый узел в W_m является непосредственным последователем некоторого узла из W_{m-1}). Отметим, что $W_M \subseteq V$ (т.е. каждый узел последнего уровня выступает листом).

Значения параметров $\sigma(w)$ и $w'(w)$ узлов w вычисляются следующим образом. Для $w \in \overline{W}$ полагаем

$$\sigma(w) = t(w), \quad w'(w) = w. \tag{2.10}$$

Отметим, что при этом будут рассчитаны значения параметров $\sigma(w)$ и $w'(w)$ для всех узлов $w \in W_M$. Пусть далее значения параметров $\sigma(w)$ и $w'(w)$ вычислены для всех узлов $w \in W_m$, $1 < m \leq M$. Для каждого узла $w \in W_{m-1} \setminus V$ рассчитываем

$$\max_{w \in Q(w)} \sigma(\overline{w}).$$

Пусть этот максимум достигается при $\overline{w} = \overline{w}_0$. Тогда полагаем

$$\sigma(w) = \sigma(\overline{w}_0) + t(w), \quad w'(w) = \overline{w}_0. \tag{2.11}$$

Перейдем к описанию приближенного алгоритма построения оптимальной расстановки.

Алгоритм 6.

Шаг 1. Включить в $\overline{W}$ все листья V дерева G.

Шаг 2. Вершины W разбить на уровни $W_1, W_2, \cdots, W_M$.

Шаг 3. Вычислить величины $\sigma(w)$ и $w'(w)$ для всех $w \in W$ по формулам (2.10), (2.11).

Выполнить шаги 4–8 $(K - |V|)$ раз.

Шаг 4. Выбрать узел $w_{i_1} \in W \setminus \overline{W}$ с максимальной величиной $\sigma(w)$, что однозначно определяет цепочку рестарта максимальной длины

$$w_{i_1} \to w_{i_2} \to \cdots \to w_{i_n}, \quad w_{i_n} \in \overline{W}, \quad w_{i_j} \notin \overline{W}, \quad j = \overline{1, n-1}, \tag{2.12}$$

которая строится с помощью значений $w'(w_{i_j})$, $j = \overline{1, n-1}$.

Шаг 5. Выполнить процедуру включения модуля контроля в цепочку (2.12). Пусть модуль контроля был присоединен к прикладному модулю w_{i_r}, $1 \leq r < n$.

Шаг 6. Включить w_{i_r} в $\overline{W}$.

Шаг 7. Положить $\sigma(w_{i_r}) = t(w_{i_r})$, $w'(w_{i_r}) = w_{i_r}$.

Шаг 8. Вычислить заново по формулам (2.10), (2.11) величины $\sigma(w_{i_j})$ и $w'(w_{i_j})$ для узлов $w_{i_{r-1}}, \ldots, w_{i_1}$ в указанной последовательности.

Таблица 1. Параметры дуг сети G

Дуга	L	U	C
$(w_i, w_j) \neq (\overline{w}, w)$	0	1	$-t(w_i)$
$(\overline{w}, w)$	1	1	0

Шаг 9. Искомой расстановкой является $\overline{W}$. Положить $T = \max\limits_{w \in \overline{W}} \sigma(w)$.

Алгоритм завершен.

Вычислительная сложность шагов 1–9 составляет $O(N)$. Следовательно, сложность алгоритма 6 есть $O(N(K - |V|))$, т.е. алгоритм 6 имеет полиномиальную вычислительную сложность. Точность алгоритма 6 определяется с помощью оценки (2.7), где T^* вычисляется по формуле (2.9).

2.5. П р о и з в о л ь н ы й г р а ф. В этом разделе предполагается, что G – произвольный ориентированный граф, не содержащий циклов. Как и раньше, к каждому прикладному модулю $w \in Q_0$ прикрепляется модуль контроля. На каждой итерации предлагаемого ниже алгоритма определяется самая длинная цепочка рестарта, в которую затем включается модуль контроля. Для нахождения такой цепочки будет использована следующая модификация алгоритма дефекта [16, 17], определяющего поток минимальной стоимости в сети.

Пусть в графе G выделены два узла $w \in W \backslash \overline{W}$ и $\overline{w} \in \overline{W}$ и требуется найти самую длинную цепочку рестарта:

$$w \to w_{i_1} \to w_{i_2} \to \cdots \to w_{i_n} \to \overline{w}, \tag{2.13}$$

где $w_{i_j} \notin \overline{W}$, $j = \overline{1, n}$. Будем рассматривать граф G как потоковую сеть с источником w и стоком $\overline{w}$. Добавим к сети G возвратную дугу $(\overline{w}, w)$. Припишем каждой дуге $(w_i, w_j) \in A$ графа G три параметра: нижнюю границу $L(w_i, w_j)$ потока по дуге, верхнюю границу $U(w_i, w_j)$ потока по дуге и стоимость $C(w_i, w_j)$ единицы потока по дуге. Значения параметров дуг сети G приведены в табл. 1.

Алгоритм дефекта, работая с параметрами, указанными в таблице, построит самый длинный ориентированный путь из w в $\overline{w}$ [16, 17]. Этот путь определяется теми дугами (кроме возвратной дуги), поток по которым равен единице. Однако такой путь, помимо $\overline{w}$, может содержать и другие узлы из $\overline{W}$, и в этом случае он не будет цепочкой рестарта. Чтобы избежать этого, предлагается использовать следующую простую модификацию алгоритма дефекта. При построении увеличивающего пути из w в $\overline{w}$ нельзя использовать узлы из $\overline{W}$. А в остальном алгоритм работает так же, как алгоритм дефекта. Перейдем к описанию приближенного алгоритма построения оптимальной расстановки.

Алгоритм 7

Шаг 1. Включить Q_0 в $\overline{W}$.

Выполнять шаги 2–4 $(K - |Q_0|)$ раз.

Шаг 2. Для каждой пары узлов w, $\overline{w}$ ($w \notin \overline{W}$, $\overline{w} \in \overline{W}$, кроме того, либо $w \in P_0$, либо w имеет одного или нескольких предшественников, если $w' \to w$, то $w' \in \overline{W}$) с помощью модифицированного алгоритма дефекта найти самый длинный путь (2.13). Этот путь является самой длинной цепочкой рестарта.

Шаг 3. Применить процедуру включения модуля контроля в построенную на шаге 3 цепочку рестарта. Пусть в результате этой процедуры модуль контроля был прикреплен к прикладному модулю w_{i_j}.

Шаг 4. Добавить w_{i_j} к $\overline{W}$.

Шаг 5. Расстановка $\overline{W}$ построена. С помощью процедуры, описанной на шаге 2, вычислить величину

$$T(\overline{W}) = \max_{w \in W} t(\Pi(w)).$$

Алгоритм завершен.

В результате работы алгоритма будет построена расстановка $\overline{W}$, являющаяся приближением к оптимальной. Вычислительная сложность отдельных шагов алгоритма следующая: шаг 1 – $O(N)$; шаг 2 – $O(|A|^2 KN)$, где $|A|$ – число дуг в сети G; шаг 3 – $O(N)$; шаг 4 – $O(1)$. Таким образом, сложность алгоритма 7 составляет $O(|A|^2 KN(K - |Q_0|))$, т.е. алгоритм 7 имеет полиномиальную вычислительную сложность. Пусть $T_{\max}$ – длина самого длинного пути в графе G. Пусть далее

$$T^* = \max(T_{\max}/(K - |Q_0|), \underset{i=1,n}{\max}(t(w_i)),$$

если $K > |Q_0|$, и $T^* = T_{\max}$, если $K = |Q_0|$. Для определения отклонения величины $T(\overline{W})$ от T_{opt} применима оценка $T(\overline{W}) - T_{opt} \le T(\overline{W}) - T^*$, которая может быть вычислена после работы алгоритма.

Заключение. Задача оптимального расположения модулей контроля в вычислительной системе реального времени сформулирована в виде минимаксной задачи. Исследованы различные структуры графа частичного порядка выполнения прикладных модулей: последовательная цепочка, несколько независимых последовательных цепочек, дерево, ориентированное от листьев к корню, дерево, ориентированное от корня к листьям, произвольный граф без циклов. Разработаны точные и приближенные алгоритмы построения оптимальной расстановки модулей контроля. Получены оценки сложности и оценки точности исследованных алгоритмов. Показано, что эти алгоритмы имеют полиномиальную вычислительную сложность.

СПИСОК ЛИТЕРАТУРЫ

1. *Танаев В.С., Гордон В.С., Шафранский Я.М.* Теория расписаний. Одностадийные системы. М.: Наука, 1984.
2. *Brucker P.* Scheduling Algorithms. Heidelberg: Springer, 2007.
3. *Лазарев А.А.* Теория расписаний. Оценка абсолютной погрешности и схема приближенного решения задач теории расписаний. М.: МФТИ, 2008.
4. *Мищенко А.В., Кошелев П.С.* Оптимизация управления работами логистического проекта в условиях неопределенности // Изв. РАН. ТиСУ. 2021. № 4. С. 86–101.
5. *Глонина А.Б., Балашов В.В.* О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов // Моделирование и анализ информационных систем. 2018. Т. 25. № 2. С. 174–192.
6. *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.
7. *Глонина А.Б.* Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем // Вестн. МГУ. Сер. 15. Вычисл. математика и кибернетика. 2020. № 3. С. 16–29.
8. *Алифанов Д.В., Лебедев В.Н., Цурков В.И.* Оптимизация расписаний с логическими условиями предшествования // Изв. РАН. ТиСУ. 2009. № 6. С. 88–93.
9. *Миронов А.А., Цурков В.И.* Минимакс в моделях транспортного типа с интегральными ограничениями // Изв. РАН. ТиСУ. 2003. № 4. С. 69–81.
10. *Миронов А.А., Цурков В.И.* Минимакс при нелинейных транспортных ограничениях // ДАН. 2001. Т. 381. № 3. С. 305–308.
11. *Grassi V., Donatiello L., Tucci S.* On the Optimal Checkpointing of Critical Tasks and Transaction-Oriented Systems // IEEE Trans. Software Eng. 1992. V. 18. № 1. P. 72–77.
12. *Coffman E., Gilbert E.* Optimal Strategies for Scheduling Checkpoints and Preventive Maintenance // IEEE Trans. Reliability. 1990. V. 39. № 1. P. 9–18.
13. *Bruno J.L., Coffman E.G.* Optimal Fault-Tolerant Computing on Multiprocessor Systems // Acta Informatica. 1997. V. 34. P. 881–904.
14. *Белый Д.В., Сушков Б.Г.* Модель организации рестартов в системах реального времени. М.: ВЦ РАН, 1996. 32 с.
15. *Гречук Б.В., Фуругян М.Г.* Алгоритмы организации рестартов в системах реального времени с произвольным графом связей. М.: ВЦ РАН, 2004. 32 с.
16. *Филлипс Д., Гарсиа-Диас А.* Методы анализа сетей. М.: Мир, 1984.
17. *Корте Б., Фиген Й.* Комбинаторная оптимизация. Теория и алгоритмы. М.: ЦНМО, 2015.