Критерий существования бесконфликтного расписания для системы строго периодических задач[1]

[1] *С.А. Зеленова <sophia@ispras.ru>*
[1,2] *С.В. Зеленов <zelenov@ispras.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Национальный исследовательский университет Высшая школа экономики, 101000, Москва, ул. Мясницкая, д. 20*

Аннотация. В критических системах выполнение жестких требований по времени взаимодействия между задачами обеспечивается строгой периодичностью запуска задач, когда каждая задача стартует через равные промежутки времени. При планировании строго периодических задач с прерываниями наиболее трудным этапом является выбор начальных стартовых точек задач. В настоящей работе предлагается новый подход к анализу расписаний, основанный на изучении раскрасок графов периодов задач и на решении систем линейных сравнений. Основным результатом является критерий существования бесконфликтного расписания для произвольного количества строго периодических задач с прерываниями на одном процессоре. Критерий позволяет либо направленно найти стартовые точки, либо быстро установить, что расписание построить невозможно.

Ключевые слова: системы реального времени; строго периодическая задача; планирование; раскраска графа; система линейных сравнений.

DOI: 10.15514/ISPRAS-2017-29(6)-10

Для цитирования: Зеленова С.А., Зеленов С.В. Критерий существования бесконфликтного расписания для системы строго периодических задач. Труды ИСП РАН, том 29, вып. 6, 2017 г., стр. 183-202. DOI: 10.15514/ISPRAS-2017-29(6)-10

1. Введение

В жестких системах реального времени, таких как автомобильная электроника, авионика, мобильная робототехника, телекоммуникация, и т.п., жизненно

[1] Поддержано грантом РФФИ, Договор № 17-01-00504\17

важно, во-первых, чтобы в каждый момент времени обрабатываемая информация была как можно более актуальна, а во-вторых, чтобы система вовремя реагировала на поступающие входные данные, причем нарушение этих требований приводит к катастрофическим последствиям. В подобных системах требуется точная регулировка взаимодействия датчиков (sensors), приводов (actuators) и функций управления обратной связью (т.е. функций обработки информации и выдачи управляющих воздействий). Для этого такие задачи запускаются строго периодически, т.е. каждая из них должна периодически стартовать через равные промежутки времени.

Несколько десятилетий назад для повышения надежности каждому процессу отводилось свое устройство. Однако к настоящему времени размеры систем настолько выросли, что подобная организация архитектуры привела бы к непомерному весу и энергопотреблению системы. Выходом здесь является разделение ресурсов, т.е. выполнение нескольких задач на одном процессоре. В связи с этим большую актуальность получает задача составления совместного расписания для нескольких процессов, выполняемых на одном или нескольких устройствах.

Планирование задач может быть динамическим (т.е. осуществляемым в реальном масштабе времени в ходе работы системы) или статическим (когда расписание составляется до запуска системы). Выполнение задач с точки зрения планирования бывает двух видов: с прерываниями (preemptive), когда выполнение задачи разрешается на время отложить для запуска другой задачи, и без прерываний (non-preemptive), когда этого делать не разрешается.

В данной статье мы затронем вопросы статического планирования строго периодических задач с прерываниями.

Строго периодическая задача задается двумя параметрами: период (время между двумя последовательными стартами задачи) и длительность (время, которое необходимо задаче для ее выполнения в пределах одного периода).

В отличие от классического случая, когда периодичность не является строгой [8], [9], построение расписания для набора строго периодических задач с прерываниями разбивается на два этапа:

- построение расписания для начальных точек данной системы периодических процессов.
- распределение остальных точек в пределах заданных периодов.

По определению строго периодической задачи все ее точки старта образуют арифметическую прогрессию. Поскольку две задачи не могут стартовать одновременно, арифметические прогрессии для разных задач не должны конфликтовать, т.е. не должны иметь общих точек. Достаточно давно известно достаточное условие бесконфликтности, состоящее в том, что начальные точки должны иметь разные остатки по модулю наибольшего общего делителя (НОД) соответствующих периодов (см., например, [11]), в частности, взаимная простота периодов немедленно влечет невозможность построения расписания.

В случае, когда задач всего две, это условие является критерием, позволяющим эффективно построить расписание. Однако, для случая произвольного количества задач подобный критерий пока не найден.

Имеющиеся подходы к проблеме планирования строго периодических задач не используют в должной мере взаимосвязи между периодами (см., например, [5], [6], [7], [10], [12]), что приводит к существенному увеличению времени поиска. Существующие алгоритмы решают проблему поиска стартовых точек либо с применением грубых эвристик [12], либо перебором с различными оптимизациями [5], [6]. Однако, переборное решение обладает тем недостатком, что в случае, если расписание построить невозможно, требуется проанализировать все возможные варианты.

В настоящей работе мы решаем проблему первого этапа — поиска системы начальных точек без конфликтов. В такой постановке мы пренебрегаем длительностью задач. Используя результаты теории графов [1], [2], [3] и теории чисел [4], мы предлагаем новый подход к анализу построения расписания для строго периодических задач, основанный на изучении структуры групп их периодов. Основным результатом является критерий существования бесконфликтного расписания для произвольного количества задач. Анализируя периоды с помощью этого критерия, можно либо направленно найти стартовые точки, либо быстро установить, что расписание построить невозможно.

Дальнейшее изложение материала статьи построено так.

В разделе 2 вводятся основные определения и приводятся элементарные условия существования бесконфликтного расписания для строго периодических задач.

В разделе 3 исследуется взаимосвязь проблемы построения бесконфликтного расписания для произвольного количества строго периодических задач и проблемы раскраски некоторых графов специального вида, построенных на основе значений периодов этих задач.

В разделе 4 вводится понятие графа делимости и исследуется вопросы совместимости раскрасок графов периодов с точки зрения возможности построения бесконфликтного расписания. Доказывается теорема, устанавливающая необходимые и достаточные условия раскраски графов для существования бесконфликтного расписания.

В разделе 5 производится обзор существующих подходов к построению расписаний для строго периодических задач.

В разделе 6 подводятся итоги статьи и намечаются пути дальнейших исследований.

2. Необходимое условие существования расписания

Задача называется *строго периодической*, если

- задача выполняется повторно с некоторой периодичностью;

- повторные запуски задачи совершаются строго через фиксированный промежуток времени, называемый *периодом*.

Длительностью строго периодической задачи называется время, отводимое на ее выполнение в рамках одного периода.

Пусть дано множество строго периодических задач, каждая из которых имеет единичную длительность. Требуется составить расписание выполнения данных задач на одном процессоре.

Введем обозначение для множества периодов: $P = \{p_i | i = 1...N\}$. Заметим, что множество P является мультимножеством, т.е. периоды, относящиеся к разным процессам в нем не сливаются, а рассматриваются как отдельные сущности, даже если они равны.

Не теряя общности можно предположить, что начальные точки всех процессов отстоят от общей точки отсчета не дальше периода соответствующего процесса. Здесь точка отсчета соответствует нулю, а каждая начальная точка соответствует какому-либо целому неотрицательному числу.

Атомарным расписанием назовем все точки одного процесса с заданным началом и заданным периодом. Атомарные расписания *конфликтуют*, когда они содержат одинаковые точки, т.е. их пересечение непусто. *Конфликтным расписанием* назовем расписание, в котором есть два конфликтующих атомарных расписания.

Утверждение 1. Если периоды p_1 и p_2 двух процессов взаимно просты, то любые расписания для этой пары процессов являются конфликтными.

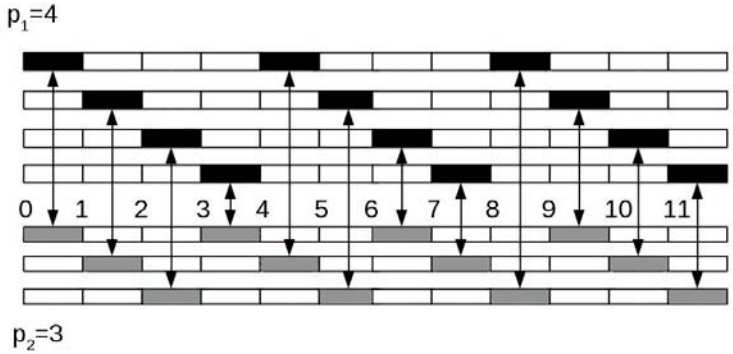

Рис. 1. Пример пары процессов, для которых не существует бесконфликтного расписания. Для каждого процесса указаны всевозможные варианты расписаний. Стрелками указаны конфликтующие позиции
Fig. 1. Example of two tasks that cannot be scheduled without conflicts. For each task, all possible variants of schedule are shown. Arrows point to conflicts.

Доказательство. Конфликтность расписания можно записать в виде равенства $t_1 + n_1 p_1 = t_2 + n_2 p_2$, где t_1, t_2 — начальные точки, а n_1, n_2 — неотрицательные целые числа. Из теории чисел известно, что для взаимно

простых чисел существует целочисленная линейная комбинация равная их наибольшему общему делителю, то есть единице: существуют целые числа k, m такие, что $kp_1 + mp_2 = 1$. Тогда $k(t_2 - t_1)p_1 + m(t_2 - t_1)p_2 = t_2 - t_1$, то есть $t_1 + k(t_2 - t_1)p_1 = t_2 + m(t_1 - t_2)p_2$. Взяв достаточно большое N, получаем $t_1 + (k(t_2 - t_1) + Np_2)p_1 = t_2 + (m(t_1 - t_2) + Np_1)p_2$, где коэффициенты $k(t_2 - t_1) + Np_2$ и $m(t_1 - t_2) + Np_1$ — положительные числа, то есть атомарные расписания имеют общую точку, а, значит, конфликтуют.□

Утверждение 2. Трансформации расписания в виде сдвига или деления точек расписания на общий множитель не меняют свойства конфликтности или бесконфликтности расписания.

Доказательство. Для сдвига утверждение очевидно. Для случая деления точек на общий множитель: две различные точки дадут разные результаты при делении на общий множитель, так что свойство бесконфликтности или конфликтности сохранится для трансформированного расписания. □

Утверждение 3. Пусть p_1 и p_2 — периоды двух задач и пусть их наибольший общий делитель равен d. Тогда, если t_1, t_2 — начальные точки этих двух процессов и атомарные расписания не конфликтуют, то t_1 и t_2 не могут иметь одинаковые остатки по модулю d.

Доказательство. Пусть $t_1 \equiv t_2 (mod\ d)$ и $t_1 < t_2$, тогда можно трансформировать расписание следующим образом: сдвинем точку отсчета в t_1 и произведем соответствующую перенумерацию (т.е. $t_{1'} = 0, t_{2'} = t_2 - t_1$). Все точки нового расписания делятся на d, так что можно произвести вторую трансформацию — поделить все точки расписания на d. При этом новые периоды $p_{1'} = p_1 / d, p_{2'} = p_2 / d$ окажутся взаимно простыми, т.к. d является наибольшим общим делителем p_1 и p_2. И, таким образом, согласно утверждению 1, трансформированное расписание будет конфликтным, что означает (согласно утверждению 2) конфликтность первоначального расписания.□

Утверждение 4. Пусть p_1 и p_2 — периоды двух процессов и пусть их наибольший общий делитель равен d. Тогда, если t_1, t_2 — начальные точки этих двух процессов и t_1 и t_2 имеют разные остатки по модулю d, то соответствующие атомарные расписания не конфликтуют.

Доказательство. Пусть $t_1 = dk_1 + r_1$ и $t_2 = dk_2 + r_2$, $r_1 \neq r_2$. Тогда наличие общей точки у атомарных расписаний можно записать в виде условия: $t_1 + n_1p_1 = t_2 + n_2p_2$. Однако, $t_1 + n_1p_1 = r_1 + dk_1 + n_1p_1 \equiv r_1 (mod\ d)$, а $t_2 + n_2p_2 = r_2 + dk_2 + n_2p_2 \equiv r_2 (mod\ d)$, отсюда, ввиду того, что $r_1 \neq r_2$, получаем невозможность указанного равенства, что и требовалось доказать.□

Следствие 1 (необходимое и достаточное условие бесконфликтности расписания). Расписание является бесконфликтным тогда и только тогда, когда в заданной системе задач для любых двух процессов с периодами p_1, p_2 начальные точки t_1, t_2 этих двух процессов имеют разные остатки по модулю наибольшего общего делителя периодов p_1, p_2.

Доказательство. Согласно утверждению 4, выполнение условия означает бесконфликтность любой пары атомарных расписаний, что влечет бесконфликтность общего расписания. С другой стороны, в бесконфликтном расписании любая пара атомарных расписаний бесконфликтна, т.е., согласно утверждению 3, начальные точки t_1, t_2 соответствующих процессов не могут иметь одинаковые остатки по модулю наибольшего общего делителя периодов.□

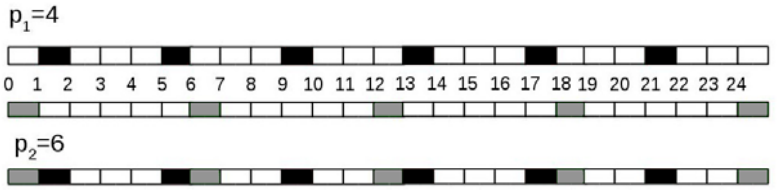

Рис. 2. Пример бесконфликтного расписания для пары процессов, у которых периоды имеют НОД не равный единице. Внизу указано общее расписание.
Fig. 2. Example of non-conflict schedule for two tasks with GCD ≠ 1. Bottom line shows the schedule itself.

Следствие 2 (необходимое условие существования бесконфликтного расписания). Если для подсистемы периодов p_i где $i = 1..k$ существует такое d, что $(p_i, p_j) = d$ для всех $i,j = 1..k$ и при этом $k > d$, то бесконфликтное расписание составить нельзя.

Доказательство. В силу следствия 1, начальные точки t_1, ... , t_k процессов с периодами p_1, ... , p_k должны иметь попарно различные остатки при делении на d, но таких остатков существует только $d < k$, так что, согласно известному принципу Дирихле, в любом расписании будут две начальные точки t_m, t_s с одинаковыми остатками по модулю d, стало быть такое расписание будет конфликтным.□

3. Задача существования бесконфликтного расписания и ее сложность

Рассмотрим полный граф G вершинами которого являются периоды задач $p \in P$, а ребро, соединяющее вершину p_1 с вершиной p_2 помечено наибольшим общим делителем периодов $d = (p_1, p_2)$. Можно выделить в графе G подграф G_d, содержащий все ребра, помеченные фиксированным числом d, и вершины, которые соединены такими ребрами.

Правильной раскраской неориентированного графа будем называть раскраску, при которой две вершины, соединенные ребром, покрашены в разные цвета. Напомним, что *хроматическим числом* графа называется минимальное число красок, для которого существует правильная раскраска.

Утверждение 5. Если хроматическое число подграфа G_d больше d, то любое расписание для данной системы задач будет конфликтным.

Доказательство. Пусть вершины, входящие в подграф G_d, отвечают периодам p_1, ... , p_k и пусть заданы начальные точки t_1, ... , t_k для задач с этими периодами. Тогда можно разбить множество t_1, ... , t_k на классы по модулю d. Таких классов всего d. Сопоставим каждому классу свою краску. Бесконфликтность расписания (согласно следствию 1) должна означать, что полученная раскраска правильная, т.е. каждое ребро в графе G_d соединяет вершины разного цвета, но, поскольку по условию хроматическое число G_d больше d, т.е. не существует правильных раскрасок с числом красок меньше или равном d, то любая раскраска в d цветов окажется неправильной, а расписание — конфликтным.□

Как нетрудно видеть, утверждение 5 является обобщением следствия 2.

Теперь зададимся следующим вопросом: какого рода графы могут встречаться в качестве подграфов G_d. Следующая теорема говорит о том, что подграф G_d может быть любым.

Теорема 1. Пусть дан неориентированный граф с n вершинами и произвольное натуральное число d. Можно разместить в вершинах данного графа систему чисел h_1, ... , h_n, такую, что $(h_i, h_j) = d$, если соответствующие вершины графа соединены ребром и $(h_i, h_j) \neq d$, если между соответствующими вершинами ребра нет.

Доказательство. Разместим в вершинах графа числа d. Далее возьмем различные простые числа α_i, $i = 1..n$ такие, что d не делится на α_i. Домножим числа в вершинах графа на эти простые числа (каждой вершине отвечает одно число α_i). Теперь любые два числа в вершинах имеют наибольший общий делитель, равный d. Далее, будем перебирать все пары вершин, между которыми нет ребра и домножать числа в данной паре на новое простое число α_s, $s > n$, которое будет отлично от всех предыдущих чисел α_1, ... , α_{s-1} и которое не является делителем d. Когда процесс закончится, мы получим систему, в которой выполнены следующие свойства:

- если между вершинами есть ребро, то числа в вершинах имеют наибольший общий делитель d, т.к. мы каждый раз домножали на различные простые числа и эти множители могут совпасть только если между вершинами ребра нет;
- если между вершинами ребра нет, то наибольший общий делитель соответствующих чисел больше d, т.к. у данных чисел имеется еще один общий множитель, не являющийся делителем d.

Итак, требуемая система чисел построена.□

Таким образом, для решения вопроса существования расписания для произвольной системы периодов необходимо уметь решать следующую задачу.

Проблема. Дан произвольный неориентированный граф. Вопрос: при каких условиях можно раскрасить вершины графа в n цветов так, чтобы любые две соседние по ребру вершины были разного цвета.

Поиск условий существования правильных раскрасок с фиксированным числом красок весьма сложен. Так, известная проблема четырех красок является утверждением, что планарность графа является достаточным условием для существования правильной раскраски в 4 цвета.

Естественным условием существования правильной раскраски в n цветов является отсутствие полных подграфов с n вершинами, однако это условие не дает гарантии того, что правильная раскраска существует. Известны примеры графов (см. рис. 3), хроматическое число которых значительно больше мощности максимального полного подграфа.

Тем не менее, существуют алгоритмы нахождения хроматического числа для данного графа, а также построения соответствующей раскраски.

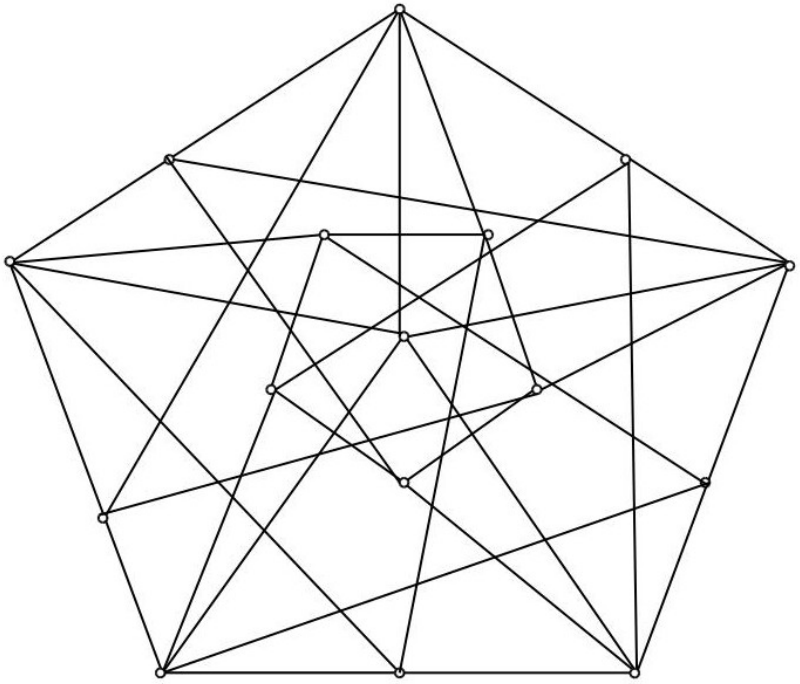

Рис. 3. Пример графа, не содержащего треугольников, с хроматическим числом, равным 5

Fig. 3. Example of triangle-free graph with chromatic number equal to 5.

4. Граф делимости и совместимость раскрасок

Пускай для всех подграфов G_d выполнено условие существования раскраски в d цветов. Естественно, возникает вопрос: существует ли в данном случае бесконфликтное расписание или нет. Следующий пример показывает, что это условие недостаточно.

Пример. Рассмотрим периоды 6, 12, 14, 18, 28, 30, 42, 154. Подграфы G_2, G_4, G_6, G_{14} изображены на рис. 4. Легко видеть, что любая правильная раскраска подграфа G_2 раскрашивает вершины 6, 12, 18, 30 в один цвет: имеется путь по ребрам через вершины 6 – 14 – 12 – 154 – 18 – 28 – 30, а т.к. цветов только два, то они должны в этой цепочке чередоваться, т.е. вершины 6, 12, 18, 30 будут окрашены в один цвет. Итак, начальные точки для вершин 6, 12, 18, 30 должны иметь одинаковую четность, а это означает, что по модулю 6 для этих начальных точек имеется только три варианта (если начальные точки нечетны, то по модулю 6 они могут иметь остатки 1, 3, 5, если четны — то остатки 0, 2, 4). Однако, взглянув на подграф G_6 мы видим, что вершины 6, 12, 18, 30 образуют полный подграф, так что по модулю 6 они должны быть раскрашены в четыре разных цвета, что невозможно, т.к. доступных цветов только три. Итак, бесконфликтное расписание в данном случае построить не удастся.

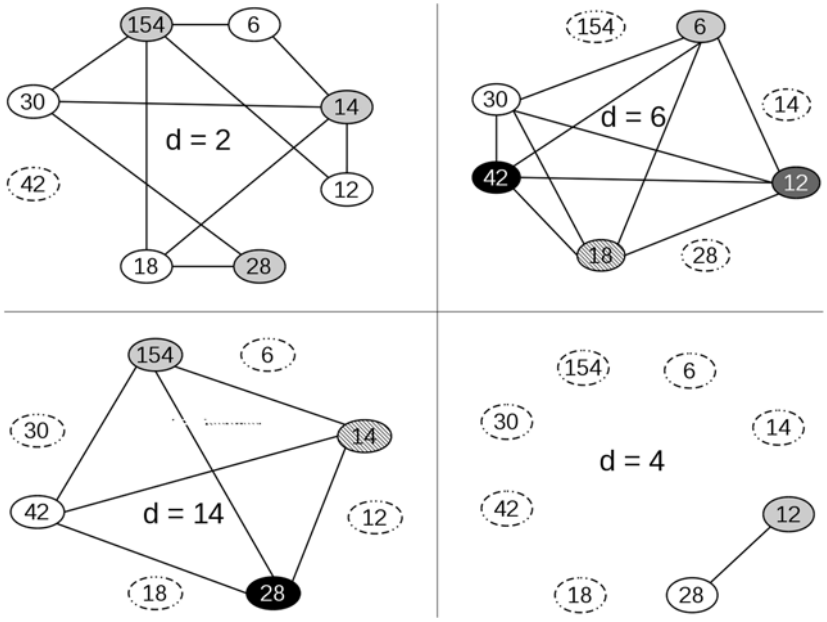

Рис. 4. Пример несовместимых правильных раскрасок графов G_d
Fig. 4. Example of incompatible proper coloring of graphs G_d

Далее, мы будем называть систему правильных раскрасок подграфов G_d *совместной*, если в этой системе возможно бесконфликтное расписание, если же такого расписания в данной системе раскрасок нет, то систему раскрасок назовем *несовместной*. Существование расписания в данной системе раскрасок означает наличие начальных точек $t_1, \ldots, t_N$ таких, что для всех d вершины подграфа G_d, покрашенные в разные цвета, имеют начальные точки с разными остатками по модулю d, а вершины, покрашенные в одинаковые цвета, имеют начальные точки с одинаковыми остатками по модулю d. Таким образом, каждому цвету соответствует ровно один остаток по модулю d.

Для дальнейшего нам понадобится еще один граф, который мы назовем *графом делимости*. Сначала построим множество вершин этого графа — множество H. Возьмем для каждой пары периодов их наибольший общий делитель d. Множество всех возможных d обозначим через D. Множество H — это замыкание множества D относительно операции взятия наибольшего общего делителя.

Множество H может быть построено стандартной процедурой замыкания:

1. В множество H добавляются все числа из D.
2. Для всех пар элементов из H вычисляются наибольшие общие делители и добавляются в множество H.
3. Шаг 2 повторяется до тех пор, пока множество H не перестанет увеличиваться.

Так как при добавлении в множество H попадают только числа, не превышающие максимального периода, то процесс построения H закончится, и количество шагов можно оценить максимальным периодом.

Множество H обладает следующими свойствами:

1. Для каждой пары чисел из H их наибольший общий делитель принадлежит H ;
2. Множество H является минимальным среди множеств, обладающих свойством 1, и содержащих D.

Построим ориентированный граф делимости G_H на основе множества H. Вершины графа G_H — элементы множества H, между вершинами h_1 и h_2 проведено ребро с началом в h_1 и концом в h_2, если

- h_1 является собственным делителем h_2
- не существует промежуточной вершины h_3, для которой h_1 делит h_3 и h_3 делит h_2.

Если имеется ребро (h_1, h_2), то h_1 будем называть родителем, а h_2 — ребенком. Две вершины соединены путем, если имеется ориентированный путь из одной вершины в другую. Вершины, из которых есть путь в вершину h, будем называть предками вершины h. По построению, всякий предок является собственным делителем своего потомка.

Утверждение 6. Если две вершины h_1 и h_2 в графе G_H имеют общего родителя d, то d — их наибольший общий делитель.

Доказательство. Действительно, если d' — наибольший общий делитель h_1 и h_2, то он присутствует в графе G_H, так как множество H замкнуто относительно операции взятия наибольшего общего делителя. Родитель d является делителем и h_1, и h_2 по построению, то есть является их общим делителем. Если d отличен от d', то d является собственным делителем d'. Т.к. h_1 и h_2 отличны друг от друга, то кто-то из них отличен от d', например, h_1. Тогда получаем, что h_1 делится на d', а d' делится на d, причем все три числа отличны друг от друга, чего не может быть по свойствам построения ребра в графе G_H.□

Рассмотрим для каждой вершины d графа G_H неориентированный граф G_d. Если d является наибольшим общим делителем какой-либо пары периодов, то G_d – граф, описанный выше (его вершины – периоды, ребра проведены там, где НОД равен d), во всех других случаях считаем граф G_d графом изолированных вершин-периодов. Пусть для каждого графа G_d дана раскраска вершин. По прежнему будем называть систему раскрасок совместной, если в этой системе можно составить бесконфликтное расписание, то есть существует такая бесконфликтная система начальных точек, что для любого графа G_d выполнено свойство: вершины, покрашенные одинаково, имеют начальные точки с одинаковыми остатками по модулю d, а вершины, покрашенные в разные цвета, имеют начальные точки с различными остатками по модулю d. Ниже мы доказываем несколько свойств, которые должны выполняться для совместимости раскрасок.

Свойство 1. В раскраске графа G_d должно участвовать не более d красок и раскраска эта должна быть правильной (т.е. если вершины соединены ребром, то они покрашены разными красками).

Доказательство. Так как остатков по модулю d ровно d, то и красок должно быть не больше d. То, что раскраска должна быть правильной, обсуждалось в предыдущем разделе.□

Свойство 2. Если в графе G_d имеется раскрашенное одним цветом множество вершин, то в графе $G_{d'}$, где d' – предок d в графе G_H это множество должно быть также одного цвета.

Доказательство. Множество вершин одного цвета означает, что им приписан один и тот же остаток по модулю d, то есть для всех периодов из этого множества начальные точки сравнимы по модулю d, но тогда они сравнимы и по модулю d', так как d' – делитель d.□

Свойство 3 (следствие свойства 2). Если в графе G_d две вершины разного цвета, то и в графе $G_{d'}$, где d' – потомок d в графе G_H, эти вершины разного цвета.

Доказательство. Если бы эти вершины в $G_{d'}$ были одинакового цвета, то, по свойству 2, в G_d они также должны были быть одного цвета, а это не так.□

Свойство 4. Если в графе G_d подмножество S вершин раскрашено в один цвет, то в графе $G_{d'}$, где d' – потомок d в графе G_H, это подмножество вершин может быть раскрашено не более чем в $\frac{d'}{d}$ цветов.

Доказательство. Как уже говорилось в доказательстве свойства 2, для всех периодов из подмножества S начальные точки сравнимы по модулю d, таким образом, остатки по модулю d' должны быть сравнимы по модулю d, а таких остатков ровно $\frac{d'}{d}$ штук (r_0, $r_0 + d$, ... , $r_0 + (\frac{d'}{d} - 1)d$, где $r_0 < d$).□

Свойство 5. Пусть d_1, ... , d_k – предки d в графе G_H, и пусть подмножество S вершин-периодов является одноцветным во всех графах G_{d_i}, $i = 1,...,k$. Тогда в графе G_d это подмножество вершин может быть раскрашено не более чем в $\frac{d}{НОК(d_1,...,d_k)}$ цветов.

Доказательство. Для всех периодов из подмножества S начальные точки сравнимы по модулю d_1, ... , d_k, то есть они (а также их остатки) сравнимы по модулю наименьшего общего кратного чисел d_1, ... , d_k, но таких остатков ровно $\frac{d}{НОК(d_1,...,d_k)}$, значит и цветов при раскрашивании S в графе G_d можно использовать не больше $\frac{d}{НОК(d_1,...,d_k)}$.□

Итак, в совместной системе все указанные свойства должны быть выполнены. Заметим, что свойство 4 является частным случаем свойства 5. Здесь оно вынесено отдельно исключительно для удобства понимания.

Заметим, что для совместности системы раскрасок необходимо, чтобы существовала нумерация красок в графах G_d остатками по модулю d такая, что система сравнений $t \equiv r(mod\ d)$ для соответствующих начальных точек t была бы разрешима для всех периодов.

Докажем несколько фактов из теории чисел, полезных для дальнейшего изложения.

Лемма 1. Если $d = ab$, где a и b — взаимно простые числа, то сравнение $x \equiv r(mod\ d)$ эквивалентно системе из двух сравнений

$$\begin{cases} x \equiv r \pmod a \\ x \equiv r \pmod b \end{cases}$$

Доказательство. Если x — решение первого сравнения, то $x = ld + r$ для некоторого l. Отсюда, имеем: $x = lab + r$, то есть сравнения $x \equiv r(mod\ a)$ и $x \equiv r(mod\ b)$ верны. Обратно, если для x выполнены оба сравнения $x \equiv r(mod\ a)$ и $x \equiv r(mod\ b)$, то $x = l_1a + r = l_2b + r$, откуда имеем, что $l_1a = l_2b$, т.е., в силу взаимной простоты чисел a и b, l_1 делится на b, так что $l_1 = l_{1'}b$, а $x = l_1a + r = l_{1'}ab + r = l_{1'}d + r$, то есть $x \equiv r(mod\ d)$.□

Лемма 2. Если $d = \alpha_{1^{k_1}} ... \alpha_{n^{k_n}}$ — разложение числа d на различные простые множители α_i , то сравнение $x \equiv r(mod\ d)$ можно заменить на систему сравнений

$$\begin{cases} x \equiv r \pmod{\alpha_1^{k_1}} \\ \dots \\ x \equiv r \pmod{\alpha_n^{k_n}} \end{cases}$$

Доказательство. Для доказательства данного утверждения достаточно несколько раз применить лемму 1.□

Лемма 3. Система сравнений $x \equiv r_i (mod\ \alpha^{k_i})$, где α — простое число, $i = 1,\dots,n$ и $k_1 < k_2 < \dots < k_n$, разрешима тогда и только тогда, когда $r_i \equiv r_j (mod\ \alpha^{k_j})$ при всех $i > j$. Если это условие выполнено, то данная система сравнений эквивалентна одному сравнению $x \equiv r_n (mod\ \alpha^{k_n})$.

Доказательство. Если система разрешима и $x = t$ — решение и $i > j$, то $t = l_i\alpha^{k_i} + r_i = l_j\alpha^{k_j} + r_j$, то есть $r_i - r_j$ делится на α^{k_j} или, что то же самое, $r_i \equiv r_j (mod\ \alpha^{k_j})$. Обратно, если $r_i \equiv r_j (mod\ \alpha^{k_j})$ при всех $i > j$ и $x = t$ — решение сравнения $x \equiv r_n (mod\ \alpha^{k_n})$, то $t = l\alpha^{k_n} + r_n \equiv r_j (mod\ \alpha^{k_j})$ для всех $j < n$, то есть t является решением всех остальных сравнений и их можно исключить, при этом множество решений не изменится.□

Лемма 4. Пусть имеется система сравнений $x \equiv r_i (mod\ d_i)$ и $d_i = \alpha_{i1}^{k_{i1}} \dots \alpha_{in_i}^{k_{in_i}}$ — разложение на простые множители. По лемме 2 можно каждое сравнение заменить на систему сравнений $x \equiv r_i (mod\ \alpha_{ij}^{k_{ij}})$. Если при этом все цепочки сравнений по модулям степеней каждого простого числа удовлетворяют условию леммы 3, то первоначальная система сравнений разрешима и может быть заменена на систему $x \equiv r_i (mod\ \alpha_{is_i}^{k_{is_i}})$, где k_{is_i} — максимальная степень простого числа α_{is_i}.

Доказательство. По лемме 3 требуемая замена может быть произведена и, таким образом, наша система сведется к системе сравнений вида $x \equiv r_i (mod\ \alpha_{ik_i})$, где α_i — различные простые числа. Но такая система имеет решение (например, по китайской теореме об остатках, т.к. разные простые числа взаимно просты), значит имеет решение и первоначальная система.□

Лемма 5. Если $a \equiv b (mod\ \alpha^s)$, а $b \equiv c (mod\ \alpha^{s'})$ для некоторого простого числа α и $s' \le s$, то $a \equiv c (mod\ \alpha^{s'})$.

Доказательство. По условию $a = \alpha^s l + b = \alpha^s l + \alpha^{s'} l' + c = \alpha^{s'} (\alpha^{s-s'} l + l') + c$.

Теорема. Если для системы раскрасок выполнены указанные выше свойства 1, 2, 5, то эта система совместна.

Доказательство. Назовем корнем графа G_H вершину, в которую не входит ни одно ребро. Поскольку у любой пары вершин из G_H есть общий предок (их наибольший общий делитель), то корень у графа G_H один.

Упорядочим все вершины графа G_H естественным образом: по величине числа d, стоящего в вершине. Такой порядок обладает одним хорошим свойством:

предок всегда идет раньше потомка, — т.к. предок является собственным делителем потомка, то он меньше потомка по величине. Самым первым в ряду вершин графа G_H относительно данного порядка оказывается корень, т.к. все остальные вершины являются его потомками.

Введем некоторые обозначения. Пусть $d_1 < d_2 < \dots < d_M$ — вершины графа G_H. Если p — некоторая раскрашенная вершина-период в графе G_d, то мы должны сопоставить ей некоторый остаток $r(p,d)$ по модулю d, так что $r(p_1,d) = r(p_2,d)$, если p_1, p_2 покрашены одной краской и $r(p_1,d) \neq r(p_2,d)$, если p_1, p_2 покрашены в разные цвета. Наша цель — так пронумеровать остатками краски, чтобы система

$$\begin{cases} x \equiv r(p, d_1) \pmod{d_1} \\ \dots \\ x \equiv r(p, d_M) \pmod{d_M} \end{cases}$$

имела решение для каждого $p \in P$.

Для построения такой нумерации будем действовать методом математической индукции.

База индукции: в корне d_1 по свойству 1 имеется не больше d_1 одноцветных множеств, и мы можем присвоить им произвольные различные остатки по модулю d_1. Очевидно, для любого остатка r по модулю d_1 существует решение сравнения $x \equiv r (mod\ d_1)$ — это сам остаток r.

Шаг индукции состоит в следующем. Предположим, что для $d_1, d_2, \dots, d_{k-1}$ нумерация красок остатками уже произведена так, что система

$$\begin{cases} x \equiv r(p, d_1) \pmod{d_1} \\ \dots \\ x \equiv r(p, d_{k-1}) \pmod{d_{k-1}} \end{cases}$$

имеет решение для каждого $p \in P$. Обозначим решение этой системы $t_{k-1}(p)$.

Построим нумерацию для красок, которыми покрашен граф G_{d_k}. Все вершины-периоды в G_{d_k} разбиваются на непересекающиеся одноцветные множества $S_1, \dots, S_m$, каждому такому множеству должен быть поставлен в соответствие свой остаток.

По свойству 2, множество S_i в любом родителе вершины d_k является одноцветным и можно этому множеству сопоставить набор цветов в родителях d_k. Два множества S_{i_1} и S_{i_2} будем называть похожими, если в каждом из родителей d_k они покрашены в одинаковый цвет. Если же в каком-то из родителей их цвет различается, то такие множества будем называть непохожими.

Рассмотрим множество S_i для некоторого i. Выберем какой-то период p, лежащий в S_i и рассмотрим решение $t = t_{k-1(p)}$ соответствующей системы сравнений. Пусть $t \equiv r (mod\ d_k)$, где $r < d_k$ — остаток от деления t на d_k. Если d_i — предок d_k, то $r \equiv r(p, d_i) (mod\ d_i)$. Действительно, $t \equiv r(p, d_i) (mod\ d_i)$,

значит $t = d_i l' + r(p, d_i) = d_k l + r$, отсюда, $r = d_i l' - d_k l + r(p, d_i) = d_i l'' + r(p, d_i)$, т.к. d_i — предок d_k, т.е. является его собственным делителем. Больше того, если m — наименьшее общее кратное всех родителей d_k, то $r + am \equiv r(p, d_i)(mod\ d_i)$ для любого целого a. Это следует из того, что m делится на d_i. Имеется ровно $\frac{d_k}{m}$ чисел вида $r + am$, попадающих в промежуток от 0 до $d_k - 1$, т.е. являющихся остатками по модулю d_k. С другой стороны, по свойству 5 имеется не больше $\frac{d_k}{m}$ множеств S_j, похожих на S_i, т.е. покрашенных в каждом из родителей d_k в тот же цвет, что и множество S_i. Действительно, если объединить все похожие на S_i множества, то в каждом родителе d_k это объединение является одноцветным, а по свойству 5 такое множество не может быть покрашено больше чем в $\frac{d_k}{m}$ цветов в графе G_{d_k}. Таким образом, можно каждому похожему на S_i множеству сопоставить остаток вида $r + am$, причем все такие остатки можно сделать разными.

Итак, мы сопоставили каждому из множеств S_i некоторый остаток по модулю d_k. Причем для разных S_i эти остатки разные: если два множества S_{i_1} и S_{i_2} похожи, то остатки у них различны по построению, а если они непохожи, то остатки будут различны, т.к. по модулю одного из родителей d_k эти остатки сравнимы с разными числами (для того из родителей, для которого множества S_{i_1} и S_{i_2} покрашены в разные цвета).

Необходимо показать, что новая система

$$\begin{cases} x \equiv r(p, d_1) \pmod{d_1} \\ \dots \\ x \equiv r(p, d_k) \pmod{d_k} \end{cases}$$

имеет решение для любого $p \in P$.

Рассмотрим данную систему сравнений. По лемме 2 можно заменить ее на систему вида

$$\begin{cases} x \equiv r(p, d_1) \pmod{\alpha_i^{s_{i1}}} \\ \dots \\ x \equiv r(p, d_k) \pmod{\alpha_i^{s_{ik}}} \end{cases}$$

где α_i — простые числа, входящие в разложение чисел d_j.

Нужно проверить не нарушилась ли совместность системы при добавлении новых сравнений. Пусть α^s — наибольшая степень некоторого простого числа α, входящая в разложение числа d_k на простые множители. Тогда к первоначальной системе добавляется сравнение $x \equiv r(p, d_k)(mod\ \alpha^s)$. Есть два случая: или s — максимальная степень из степеней α, входящих в d_i или же эта степень не максимальная.

Пусть s — максимальная степень α и пусть некоторое $d = d_j$, $j < k$ имеет в своем разложении степень $\alpha^{s'}$, $s' \le s$. Тогда имеется общий предок d' вершин d

и d_k, который также имеет в своем разложении степень $\alpha^{s'}$. Пусть d'' — родитель d_k, такой, что d' делит d'', тогда d'' делится на некоторую степень $\alpha^{s''}$, так что $s' \le s'' \le s$. По построению, $r(p, d_k) \equiv r(p, d'')(mod\ d'')$, то есть $r(p, d_k) \equiv r(p, d'')(mod\ \alpha^{s''})$. По предположению индукции, $r(p,d) \equiv r(p, d')(mod\ \alpha^{s'})$ и $r(p, d'') \equiv r(p, d')(mod\ \alpha^{s'})$. Отсюда, по лемме 5, $r(p, d_k) \equiv r(p, d')(mod\ \alpha^{s'})$ и $r(p, d_k) \equiv r(p,d)(mod\ \alpha^{s'})$, таким образом условие леммы 3 не нарушается при добавлении нового сравнения.

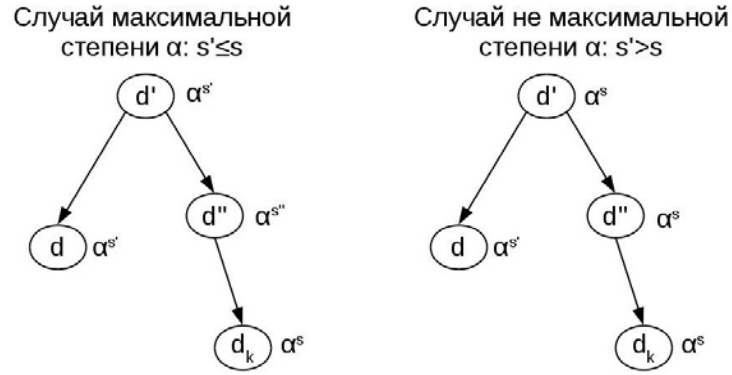

Рис. 5. Иллюстрация распределения степеней α в разных случаях
Fig. 5 Distribution of degrees of α in different cases.

Если же s — не максимальная степень α, то имеется $d = d_j$, $j < k$, имеющее в своем разложении степень $\alpha^{s'}$, $s' > s$. Аналогично предыдущему случаю, имеется общий предок d' вершин d и d_k, который имеет в своем разложении степень α^s. Пусть d'' — родитель d_k, такой, что d' делит d'', тогда d'' делится на степень α^s. По предположению индукции, $r(p,d) \equiv r(p, d')(mod\ \alpha^s)$. С другой стороны, по построению $r(p, d_k) \equiv r(p, d'')(mod\ d'')$, то есть $r(p, d_k) \equiv r(p, d'')(mod\ \alpha^s)$, и, таким образом, новое сравнение эквивалентно одному из уже имевшихся, так что условие леммы 3 снова выполнено.

Итак, при добавлении новых сравнений условие леммы 3 выполняется, то есть, по лемме 4 новая система сравнений будет иметь решение.

Осталось показать, что расписание, в котором решения $t_M(p)$ являются начальными точками будет бесконфликтным. Это вытекает из того, что $t_M(p) \equiv r(p,d)(mod\ d)$, и для p_1, p_2 с наибольшим общим делителем d в графе G_d вершины p_1, p_2 разного цвета, то есть, по построению, остатки $r(p_1,d)$, $r(p_2,d)$ также различны. То есть $t_M(p_1)$ и $t_M(p_2)$ имеют разные остатки по модулю $d = (p_1, p_2)$, и, согласно следствию 1, расписание бесконфликтно.□

Из доказательства теоремы видно, что для совместности системы раскрасок достаточно выполнения свойства 5 для родителей вершины d, а не для

произвольной системы предков этой вершины, так что окончательный результат можно сформулировать так:

Теорема о существовании бесконфликтного расписания. Для системы периодов P существует бесконфликтное расписание тогда и только тогда, когда существует система раскрасок графов G_d для всех $d \in H$ такая, что

- В раскраске графа G_d используется не более d красок и раскраска эта правильная.
- Если в графе G_d имеется раскрашенное одним цветом множество вершин, то в графе $G_{d'}$, где d' – предок d в графе G_H это множество также одного цвета.
- Пусть $d_1, \dots, d_k$ – родители d в графе G_H, и пусть подмножество S вершин-периодов является одноцветным во всех графах G_{d_i}, $i = 1,\dots,k$. Тогда в графе G_d это подмножество вершин раскрашено не более чем в $\frac{d}{НОК(d_1,\dots,d_k)}$ цветов.

5. Обзор существующих подходов к построению расписаний для строго периодических задач

В работах [7], [10] исследуется случай планирования строго периодических задач без прерываний. Доказываются критерии бесконфликтности для случая двух задач на основе НОД их периодов. Также приводится ряд громоздких достаточных условий бесконфликтности в общем случае на основе рассмотрения НОД всех периодов.

Отметим, что НОД всех периодов является слишком грубой характеристикой. Например, для задач с периодами 6, 10, 15 расписание построить возможно, хотя их НОД = 1.

В работах [5], [6], [11], [12] исследуется случай планирования строго периодических задач с прерываниями.

В работе [11] доказаны условия невозможности построения расписания:

- какие-то два периоды взаимно-просты;
- НОД периодов каких-то двух задач делит разность их стартовых точек.

Кроме того, приводится громоздкое достаточное условие невозможности построения расписания в случае, когда для данных задач дополнительно введено отношение «предшествования».

В работе [12] предлагается эвристический алгоритм построения расписаний с условием, что выход из прерывания требует ненулевое время. Задачи планируются последовательно по мере возрастания их периодов, причем в качестве точки старта для очередной задачи выбирается наименьшая из гарантированно свободных точек. Если этого сделать не удалось, то считается,

что расписание построить невозможно. Отметим, что это вообще говоря не так, поскольку алгоритм не перебирает другие возможности для выбора стартовой точки. Тем не менее, алгоритм работает в случае, когда количество задач сравнительно невелико, а попарные НОД их периодов достаточно большие — тогда множество гарантированно свободных точек не успевает стать пустым к моменту планирования последней задачи.

В работах [5], [6] предлагаются переборные (с некоторыми оптимизациями) алгоритмы построения расписаний с целью минимизировать суммарное количество прерываний (при этом сами прерывания считаются «бесплатными»). Собственно перебор используется для поиска стартовых точек.

Отметим, что во всех приведенных алгоритмах (в т.ч. «переборной» модификации алгоритма из [12]) наличие реального конфликта в расписании можно установить только перебрав все возможные варианты.

6. Заключение

В настоящей работе предложен новый подход к анализу возможности построения расписания для строго периодических задач. На основе предложенного представления структуры группы периодов в терминах теории графов доказан критерий существования бесконфликтного набора стартовых точек для произвольного количества задач. Полученный критерий позволяет либо направленно найти стартовые точки, либо быстро установить, что расписание построить невозможно.

Результат настоящей работы позволяет продолжить исследования проблемы построения бесконфликтных расписаний и в то же время указывает на сложность этой проблемы в общем случае. В рамках данной проблемы возникают, в частности, следующие направления для дальнейших исследований:

- создание алгоритмов построения расписаний (в том числе для задач с произвольными длительностями);
- вопрос оптимизации распределения задач по нескольким устройствам с заданными характеристиками;
- вопрос изменения расписания при добавлении новых задач;
- выбор подходящих характеристик (например, периодов) задач, если есть возможность менять эти характеристики в некоторых пределах;
- составление расписания при наличии дополнительных требований, таких как синхронизация задач или дополнительные ограничения на время пересылки сообщений.

Исследованию этих и других смежных вопросов мы посвятим наши следующие работы.

Список литературы

[1] Зыков А.А. Основы теории графов. М: Вузовская книга, 2004.
[2] Кристофидес Н. Теория графов. Алгоритмический подход. М: Мир, 1978.
[3] Оре О. Теория графов. М: Наука, 1980.
[4] Виноградов И.М. Основы теории чисел. М.-Л.: Гостехиздат, 1952.
[5] Зеленов С.В. Планирование строго периодических задач в системах реального времени. Труды ИСП РАН, том 20, 2011 г., с. 113-122.
[6] Третьяков А. Автоматизация построения расписаний для периодических систем реального времени. Труды ИСП РАН, том 22, 2012 г., стр. 375-400. DOI: 10.15514/ISPRAS-2012-22-20.
[7] Kermia O., Sorel Y. Schedulability Analysis for Non-Preemptive Tasks under Strict Periodicity Constraints. Proceedings of the 2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2008, p. 25-32.
[8] Liu C.L., Layland J.W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. J. ACM, 1973, No 20, p. 46-61.
[9] Liu J.W.S.W. Real-Time Systems. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edn. 2000.
[10] Marouf M., Sorel Y. Schedulability conditions for non-preemptive hard real-time tasks with strict period. Proceedings of 18th International Conference on Real-Time and Network Systems, RTNS'10, 2010, p. 50-58.
[11] Yomsi P.M., Sorel Y. Non-Schedulability Conditions for Off-line Scheduling of Real-Time Systems Subject to Precedence and Strict Periodicity Constraints. Proceedings of 11th IEEE International Conference on Emerging technologies and Factory Automation, ETFA'06, WIP, Prague, Czech Republic, September 2006.
[12] Yomsi P.M., Sorel Y. Schedulability Analysis for non Necessarily Harmonic Real-Time Systems with Precedence and Strict Periodicity Constraints using the Exact Number of Preemptions and no Idle Time. Proceedings of the 4th Multidisciplinary International Scheduling Conference, MISTA'09, Dublin, Ireland, August, 2009.

Non-conflict scheduling criterion for strict periodic tasks

[1] *S.A. Zelenova <sophia@ispras.ru>*
[1,2] *S.V. Zelenov <zelenov@ispras.ru>*
[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *National Research University Higher School of Economics (HSE) 20 Myasnitskaya Ulitsa, Moscow, 101000, Russia*

Abstract. In the paper, we address mission critical systems, such as automobile, avionic, mobile robotic, telecommunication, etc. Such systems must meet hard real-time constraints in order to avoid catastrophic consequences. To meet the real-time constraints, strict periodicity is used (i.e. for any periodic task, time between release points is constant). Sensors, actuators and feedback control functions are typical examples of strict periodic tasks. We study a monoprocessor preemptive scheduling problem for arbitrary number of strict periodic tasks. In this context, we focus on the following problem: how to find non-conflict set of task release points (i.e. sequences of instance release points for different tasks must not intersect). First, as a preliminaries, we introduce some fundamental definitions and prove several elementary schedulability conditions. Next, we investigate the correlation between the scheduling problem and a graph coloring problem for graphs of some special kinds. The graphs under consideration are built on the basis of the tasks' period values. We introduce a notion of divisibility graph for tasks' periods, and study compatibility of graphs' coloring with respect to the schedulability problem. At last, we prove a theorem that provides necessary and sufficient graph coloring conditions for schedulability of given strict periodic tasks. This theorem allows either to find non-conflict set of task release points directly, or to determine quickly that scheduling is impossible.

Keywords: real-time system; strict periodic task; scheduling; graph coloring; system of linear congruences.

DOI: 10.15514/ISPRAS-2017-29(6)-10

For citation: Zelenova S.A., Zelenov S.V. Non-conflict scheduling criterion for strict periodic tasks. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 6, 2017. pp. 183-202 (in Russian). DOI: 10.15514/ISPRAS-2017-29(6)-10

References

[1] A. A. Zykov. Fundamentals of Graph Theory. BCS Associates, 1990.
[2] N. Christofides. Graph Theory. An Algorithmic Approach. Academic Press, 1975.
[3] O. Ore. Theory of graphs. American Mathematical Society, 1962.
[4] I.M. Vinogradov. Elements of Number Theory. Dover Publications Inc. 1954.
[5] S.V. Zelenov. Scheduling of Strictly Periodic Tasks in Real-Time Systems. Trudy ISP RAN/Proc. ISP RAS, vol. 20, 2011, pp. 113-122 (in Russian).
[6] A.V. Tretyakov. Automation of scheduling for periodic real-time systems. Trudy ISP RAN/Proc. ISP RAS, vol. 22, 2012, pp. 375-400 (in Russian). DOI: 10.15514/ISPRAS-2012-22-20.
[7] Kermia O., Sorel Y. Schedulability Analysis for Non-Preemptive Tasks under Strict Periodicity Constraints. Proceedings of the 2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2008, p. 25-32.
[8] Liu C.L., Layland J.W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. J. ACM, 1973, No 20, p. 46-61.
[9] Liu J.W.S.W. Real-Time Systems. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edn. 2000.
[10] Marouf M., Sorel Y. Schedulability conditions for non-preemptive hard real-time tasks with strict period. Proceedings of 18th International Conference on Real-Time and Network Systems, RTNS'10, 2010, p. 50-58.
[11] Yomsi P.M., Sorel Y. Non-Schedulability Conditions for Off-line Scheduling of Real-Time Systems Subject to Precedence and Strict Periodicity Constraints. Proceedings of 11th IEEE International Conference on Emerging technologies and Factory Automation, ETFA'06, WIP, Prague, Czech Republic, September 2006.
[12] Yomsi P.M., Sorel Y. Schedulability Analysis for non Necessarily Harmonic Real-Time Systems with Precedence and Strict Periodicity Constraints using the Exact Number of Preemptions and no Idle Time. Proceedings of the 4th Multidisciplinary International Scheduling Conference, MISTA'09, Dublin, Ireland, August, 2009.