© 2012 г. П.Е. ШЕСТОВ
(МГУ им. М.В. Ломоносова, факультет Вычислительной математики и кибернетики, Лаборатория Вычислительных комплексов)

АЛГОРИТМ НА ОСНОВЕ МЕТОДА ВЕТВЕЙ И ГРАНИЦ СОВМЕСТНОГО ПЛАНИРОВАНИЯ ВЫЧИСЛЕНИЙ И ОБМЕНОВ В СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

В данной работе исследуется задача построения совместных расписаний выполнения прикладных задач и передачи сообщений. Данная задача возникает при проектировании информационно-управляющих систем реального времени. Для решения предлагается алгоритм, основанный на методе ветвей и границ. Приведены результаты его экспериментального исследования.

BRANCH-AND-BOUND ALGORITHM FOR COMBINED SCHEDULING OF COMPUTATIONS AND DATA EXCHANGES IN REAL-TIME SYSTEMS / P.E. Shestov (Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics, Computer Systems Lab, Leninskie Gory, 1, building 52, room 764, Moscow 119992, Russia, E-mail: osmin@lvk.cs.msu.su). In this article problem of combined scheduling of applications and data exchanges is addressed. This problem arises in the design of real-time management information systems. Branch-and-bound algorithm is proposed. Experimental results are presented.

1. Введение

Одним из важнейших требований к функционированию информационно–управляющих систем реального времени (ИУС РВ) является соблюдение директивных сроков выполнения прикладных программ. При нарушении директивных сроков, ИУС РВ теряет свою работоспособность. При разработке и отладке прикладного программного обеспечения для каждой версии программного обеспечения ИУС РВ необходимо строить расписание выполнения прикладных программ и расписание обмена данными для проверки возможности выполнения ограничений реального времени. При построении расписаний необходимо учитывать технологические ограничения ИУС РВ, обусловленные особенностями используемых аппаратных и системных программных средств. Рассматриваемая в работе задача совместного планирования вычислений и обменов возникает при модификации ИУС РВ (например, при добавлении новых подсистем), а также при разработке новых ИУС РВ на базе существующих.

В данной работе задача совместного планирования вычислений и обменов рассматривается для ИУС РВ, в которой в качестве среды обмена используется канал с централизованным управлением. К основным особенностям задачи можно отнести:

— в качестве исходных данных задаются набор прикладных программ, подлежащих планированию, и набор сообщений, посредством которых прикладные программы взаимодействуют друг с другом и подсистемами, изменение которых недопустимо;

— время передачи сообщения зависит от расписания выполнения прикладных программ;

— присутствуют технологические ограничения на корректность расписания обменов, а не только ограничения на передачу каждого сообщения в рамках его директивного интервала.

2. Задача совместного планирования вычислений и обменов.

Исходными данными для задачи совместного планирования вычислений и обменов являются набор прикладных программ (работ) и набор сообщений, посредством передачи и приема которых работы взаимодействуют друг с другом и подсистемами. Для каждой работы задан директивный интервал, в рамках которого она должна быть выполнена, и набор вычислительных модулей, на которые работа может быть размещена.Для каждого сообщения задан директивный интервал, в рамках которого оно должно быть передано. Прерывание выполнения работ и передачи сообщений недопустимо. ИУС РВ рассматривается как система, состоящая из набора вычислительных модулей и подсистем, подключенных к единой среде передачи данных.Каждая подсистема может быть представлена как набор работ (в дальнейшем работы-подсистемы), исполняющихся на отдельном вычислительном модуле, на который недопустимо назначение работ из исходного набора. Для множества работ-подсистем известно расписание. Они не подлежат планированию, но участвуют в передаче и приеме сообщений наряду с другими работами. Задача совместного планирования вычислений и обменов заключается в построении согласованных статических расписаний выполнения исходно заданных работ и сообщений.

Полная формальная постановка проблемы приведена в [1]. Здесь же приведем её краткий вариант и введем необходимые обозначения для описания алгоритма решения задачи. Исходный набор работ и сообщений будем представлять ориентированным ациклическим графом (возможно, несвязным) с двумя типами вершин $G = (\overline{W} \cup M,\ \prec)$. Первый тип вершин соответствует работам (в том числе работам-подсистемам), второй — сообщениям. Дугами графа задаётся частичный порядок на множествах работ и сообщений. Пример графа приведен на рисунке 1. Квадратами отмечены вершины-работы, кругами — вершины-сообщения. Дополнительно вершины, соответствующие работам-подсистемам, отмечены пунктиром. Работы-подсистемы не подлежат планированию, но участвуют в передаче сообщений наряду с другими работами. Каждая вершина-сообщение передаёт данные от вершины-работы, являющейся её предшественником, вершинам-работам, являющимся её преемниками. Если у вершины-работы среди предшественников есть вершины-работы (как, например, вершины-работы $1 - 3$), то все они вместе с исходной вершиной-работой должны быть запланированы на один и тот же вычислительный модуль, а обмен данными происходит через внутреннюю память этого модуля. При этом каждой вершине-сообщению инцидентны ровно одна входящая дуга и не менее одной исходящей дуги. Другими словами мы предполагаем, что данные, передаваемые в сообщении, формируются специальной работой, которая может брать слова данных от других работ, выполняющихся на этом же вычислительном модуле. Такой способ используется во многих ИУС РВ, в которых есть каналы с централизованным управлением.

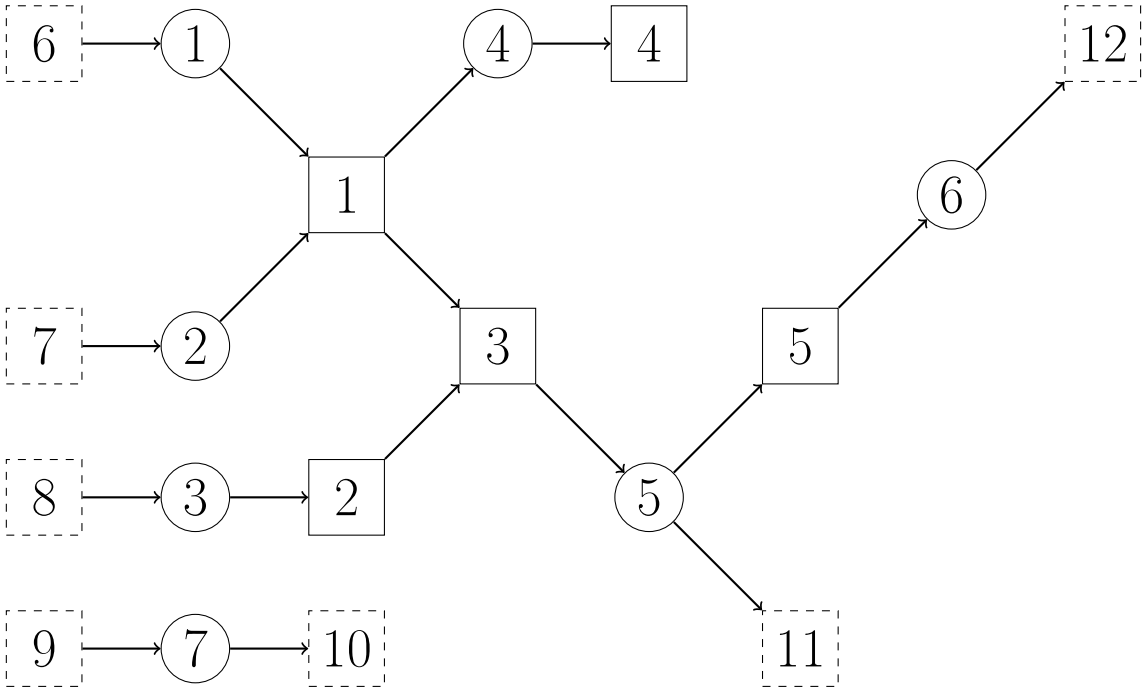

Рис. 1. Граф зависимости работ и сообщений

Обозначим через $P = \{1, \ldots, n_P\}$ множество вычислительных модулей, $W = \{1, \ldots, i, \ldots, n_W\}$ — множество работ из исходно заданного набора, которые подлежат планированию ($W \subseteq \overline{W}$), $M = \{1, \ldots, j, \ldots, n_M\}$ — множество всех сообщений.

Для каждой вершины-работы из множества W известны:

s_i^w — директивный срок начала выполнения работы (левый директивный срок);

f_i^w — директивный срок завершения выполнения работы (правый директивный срок);

$P_i \subseteq P$ — набор вычислительных модулей, на которые допустимо назначение этой работы для выполнения.

Также известна функция $dw : W \times P \to \mathbb{N}$, определяющая длительность выполнения заданной работы на заданном вычислительном модуле. Здесь и далее $\mathbb{N}$ обозначает множество натуральных чисел. Аналогично для каждой вершины-сообщения из множества M известны:

s_j^m — директивный срок начала передачи сообщения (левый директивный срок);

f_j^m — директивный срок завершения передачи сообщения (правый директивный срок).

Если среди предшественников или преемников вершины-сообщения есть работы из множества W, то длительность передачи сообщения зависит от того, на какие вычислительные модули эти работы будут назначены. Более того, если все предшественники и преемники вершины-сообщения будут запланированы на один вычислительный модуль, то сообщение вообще не требуется передавать по среде передачи данных. В состав исходных данных входит функция, позволяющая определить длительность передачи сообщения, однако формальное её представление будет введено ниже, так как требует введения понятия привязки работ к вычислительным модулям.

Введём дополнительные обозначения. Для каждой вершины-работы i в графе исходных данных примем:

M_i^{in} — множество вершин-сообщений — непосредственных предшественников;

M_i^{out} — множество вершин-сообщений — непосредственных потомков;

W_i^{Win} — множество вершин-работ — непосредственных предшественников;

W_i^{Wout} — множество вершин-работ — непосредственных потомков.

Для каждой вершины-сообщения j в графе исходных данных запишем:

w_j^{Min} — вершина-работа — непосредственный предшественник;

W_j^{Mout} — множество вершин-работ — непосредственных потомков.

Определение 1. Расписание выполнения работ H_W представляет собой множество троек вида <работа, номер вычислительного модуля на котором выполняется работа, планируемое время старта работы>.

Обозначим $\hat{s}_i^w$ время начала выполнения работы i, а промежуток времени $[\hat{s}_i^w; \hat{s}_i^w + dw\,(i, p_i)]$ будем называть интервалом выполнения работы i. Для простоты, когда известно (или не важно), о каком расписании работ идёт речь, будем обозначать длительность выполнения работы i в нём как dw_i.

Определение 2. Расписание H_W называется корректным, если выполнены следующие ограничения корректности (ограничения для многоприборных систем без прерывания работ):

1) *каждая работа из набора включена не более чем в одну тройку;*

2) *требование реального времени (каждая работа выполняется в рамках своего директивного интервала);*

3) *на одном и том же вычислительном модуле в каждый момент времени может выполняться только одна работа;*

4) *каждая работа должна быть запланирована только на допустимый для неё вычислительный модуль;*

Определение 3. Расписание H_W называется полным, если каждая работа из набора W включена в расписание. Множество всех расписаний работ (в том числе некорректных и неполных) обозначим через $\{H_W\}$.

Определение 4. Привязкой работ A к вычислительным модулям назовём множество пар <работа, вычислительный модуль>.

Будем говорить, что расписанию H_W соответствует привязка A, если для каждой тройки в расписании есть соответствующая пара в привязке и наоборот. Сужением привязки на множество W' будем называть такое её максимальное подмножество, что все его работы содержатся в множестве W'.

Определение 5. Расписание передачи сообщений H_M представляет собой множество пар вида <сообщение, время старта>.

Множество всех расписаний передачи сообщений из набора M обозначим через $\{H_M\}$. Функция, позволяющая определить длительность передачи сообщений, выглядит следующим образом: $dm : M \times \{A\} \to \mathbb{N}$. При этом для каждого конкретного сообщения $i \in M$ на значение его длительности влияет привязка не всех работ, а лишь её сужения на множество принимающих и передающих работ. Поскольку длительность передачи сообщений зависит от привязки работ к вычислительным модулям, то рассматривать корректность расписания передачи сообщений возможно только в контексте расписания выполнения работ и соответствующей ему привязке работ. Обозначим

$\hat{s}_i^m$ время начала передачи сообщения i, а промежуток времени $[\hat{s}_i^m; \hat{s}_i^m + dm\,(i, A)]$ будем называть интервалом передачи сообщения i. Для простоты, когда известно (или не важно), в контексте какого расписания работ идёт речь, будем обозначать длительность передачи сообщения i в нём как dm_i.

Определение 6. Расписание H_M называется корректным, если выполнены следующие ограничения корректности:

1) *каждое сообщение из набора включено не более чем в одну пару;*

2) *требование реального времени (каждое сообщение должно быть передано в рамках своего директивного срока);*

3) *в каждый момент времени может передаваться только одно сообщение;*

4) *ограничения корректности, обусловленные технологическими требованиями к обмену данными.*

Определение 7. Расписание H_M называется полным, если каждое сообщение из набора M включено в расписание.

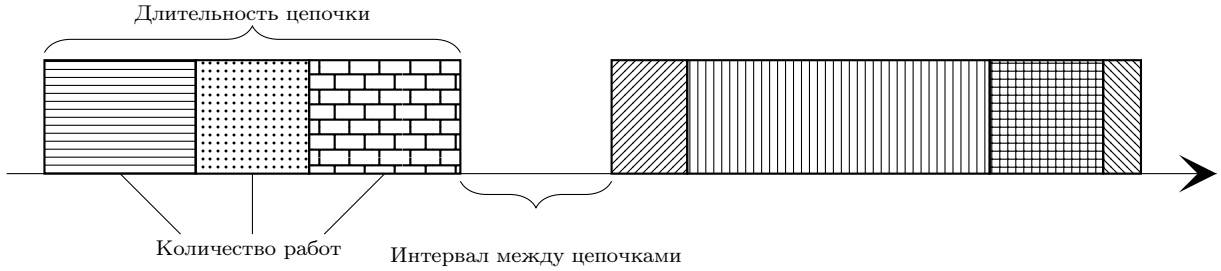

Рис. 2. Технологические требования к расписанию сообщений

Поясним содержание четвертого пункта из определения 6. Особенности архитектуры аппаратных и программных средств канала с централизованным управлением, используемого во многих ИУС РВ, определяет следующую организацию информационного обмена. Каждое передающееся сообщение однозначно относится к некоторому множеству сообщений, называемому цепочкой, которая является последовательностью сообщений, передающихся непрерывно друг за другом (см. рис. 2). При этом накладываются следующие ограничения:

1) суммарная длительность передачи сообщений в цепочке не должна превышать заданного значения r_{mct} (максимальная длительность цепочки сообщений);

2) число сообщений в цепочке не должно превышать заданного значения r_{mcc} (максимальное количество сообщений в цепочке);

3) между цепочками сообщений должен быть свободный промежуток времени, длительность которого должна быть не меньше заданного значения r_{mi} (минимальный интервал между цепочками).

Определение 8. Совместное расписание выполнения работ и передачи сообщений представляет собой пару расписаний вычислений и обменов $H = (H_W, H_M)$.

Определение 9. Совместное расписание называется полным, если являются полными оба расписания вычислений и обменов.

Определение 10. Совместное расписание H называется корректным, если выполнены следующие ограничения корректности:

1) *ограничения корректности для расписания выполнения работ;*

2) *ограничения корректности для расписания передачи сообщений;*

3) *в расписании передачи сообщений запланированы все сообщения, передающие данные, необходимые для выполнения запланированных работ;*

4) *в расписании выполнения работ запланированы все работы, формирующие данные, передающиеся в запланированных в расписании обменов сообщениях;*

5) *в расписании выполнения работ все работы, передающие данные запланированным работам через внутреннюю память вычислительных модулей, запланированы на тот же вычислительный модуль, что и получатели данных;*

Задача 1. Задача построения совместного расписания выполнения работ и передачи сообщений заключается в построении расписания, которое содержит максимально возможное число вершин графа G и удовлетворяет условиям корректности.

Суммарное количество работ и сообщений в расписании H обозначим как $|H| = |H_W| + |H_M|$. В качестве исходных данных задачи 1 заданы:

— набор вычислительных модулей P, на которые возможно планирование работ;

— граф G исходных данных с атрибутами работ и сообщений;

— функция $dw : W \times P \to \mathbb{N}$, вычисляющая время выполнения работ;

— функция $dm : M \times \{A\} \to \mathbb{N}$, определяющая время передачи сообщений;

— конкретные значения характеристик канала с централизованным управлением: r_{mct}, r_{mcc} и r_{mi}.

К основным особенностям данной задачи, отличающим ее от других задач построения расписаний, можно отнести:

— в графе исходных данных содержатся вершины двух типов: вершины-сообщения и вершины-работы, которые соответствуют прикладным задачам;

— вершины-работы могут быть двух типов: работы, выполнение которых должно быть запланировано на заданном наборе вычислительных модулей, и работы, которые не подлежат планированию и выполняются на отдельном вычислительном модуле, на который недопустимо назначение других работ;

— время передачи сообщения зависит от расписания выполнения работ, а именно от того, на какие вычислительные модули размещены передающая и принимающие работы;

— присутствуют ограничения корректности сразу на все расписание сообщений, а не только временные ограничения на каждое сообщение в отдельности;

— ищется наиболее полное корректное расписание, тогда как во многих задачах планируются все работы и сообщения, таким образом, чтобы минимизировать нарушения временных ограничений

3. Алгоритм решения задачи

В [2] был проведен анализ возможных подходов к решению задачи и приведены постановки и методы решения двух наиболее близких задач (а именно [3] и [4]). Основные проблемы, которые препятствуют применению результатов этих работ следующие:

— нет возможности учитывать технологические ограничения;

— другой критерий оптимальности (в обоих работах в том или ином виде минимизируется запаздывание).

Учет этих различий означал бы построение нового алгоритма без учета результатов [3] и [4]. Ниже будет описан алгоритм решения задачи совместного планирования, основанный на методе ветвей и границ [5].

3.1. Отношение сравнения на множестве работ и сообщений

Перед описанием основных операций алгоритма введём на множестве работ и сообщений отношение сравнения для заданного расписания H, которое обозначим $<_H$. Для двух незапланированных заданий (задание — это работа или сообщение) i и j верно, что $i <_H j$ тогда и только тогда, когда минимально возможное время, на которое возможно разместить задание i в расписание H без нарушений ограничений корректности, меньше такого же времени для задания j. Если же эти времена одинаковы, то соотношение верно тогда и только тогда, когда директивный срок завершения работы i меньше директивного срока завершения работы j. Минимально возможное время для размещения задания i является максимумом из следующих величин: директивный срок начала выполнения/передачи (f_i^w или f_i^m); время завершения непосредственных предшественников, для сообщений это $\hat{s}^w_{w_i^{Min}} + dw_{w_i^{Min}}$, для работ —

$$c_i = \max\left(\max_{j \in W_i^{Win}}\left(\hat{s}_j^w + dw_i\right), \max_{j \in M_i^{in}}\left(\hat{s}_j^m + dm_i\right)\right);$$

и время завершения последней работы или сообщения, выполняющего или передающегося на этом же устройстве. При этом, если сообщение не может быть запланировано без нарушений технологических ограничений в конец последней цепочки сообщений, то к времени завершения последнего сообщения необходимо добавить r_{mi}. Так же примем, что если для задания i все предшественники запланированы в расписании H, а для задания j — нет, то $i <_H j$. Работу (или сообщение) i будем называть минимальной для расписания H, если любая другая незапланированная работа или любое другое незапланированное сообщение не превосходит i для H: $(\forall j \in W : \neg(j <_H i)) \wedge (\forall j \in M : \neg(j <_H i))$.

3.2. Общее описание алгоритма и основных операций

Каждая вершина V дерева решений представляет собой размещение работ A^V (обозначим W^V множество этих работ) по вычислительным модулям (нелистовая вершина — неполное размещение) и множество сообщений, отправителями которых являются работы из W^V или работы-подсистемы, $M^V = \left\{i \in M \,\middle|\, \left(\exists j \in W^V : i \in M_j^{out}\right) \vee \left(\exists j \in \overline{W} \setminus W : i \in M_j^{out}\right)\right\}$. Т.е. вершине V соответствует множество работ W^V, каждой из которых приписан вычислительный модуль, на котором она будет выполняться, и множество сообщений — непосредственных потомков этих работ или работ-подсистем. Начальной вершине соответствует пустое размещение работ и множество сообщений — непосредственных потомков работ-подсистем. Так же каждой вершине при порождении присваивается порядковый номер такой, что вершины с меньшими номерами созданы раньше, чем вершины с большими.

Для каждой вершины вычисляется как оценка снизу, так и оценка сверху. Для оценки вершины V строится наиболее полное корректное совместное расписание H^V для множества работ W^V, для каждой из которых допустим только один вычислительный модуль, и набора сообщений M^V. Алгоритм построения такого расписания будет приведен ниже. Оценка снизу равна количеству работ и сообщений, которые были запланированы в расписании для данной вершины: $V_{min} = \left|H^V\right|$. Оценка сверху равна общему числу работ и сообщений в исходных данных задачи минус число работ и сообщений, которые не удалось запланировать в корректном расписании для данной вершины V: $V_{max} = |W| + |M| - \left(\left|W^V\right| + \left|M^V\right| - \left|H^V\right|\right)$.

Для ветвления выбирается вершина V с наибольшей верхней оценкой, если таких несколько, то среди них выбирается вершина с наибольшей нижней оценкой, если и таких несколько, то вершина с наименьшим порядковым номером.

Алгоритм 1 (Процедура ветвления).

1. Выбрать минимальную для H^V работу $i \in W \setminus W^V$.

2. Составить множество вычислительных модулей P_i^V, добавив в него все допустимые для i вычислительные модули: $P_i^V = P_i$.

3. Выбрать из P_i^V вычислительный модуль p с наименьшим числом запланированных в H^V работ.

4. Добавить в список вершин потомка вершины V, в котором работа i размещена на вычислительный модуль p и в который добавлены все сообщения из M_i^{out} — непосредственные потомки i.

5. Удалить p из P_i^V.

6. Если множество P_i^V непусто, то перейти на шаг 3, иначе — на шаг 7.

7. Удалить вершину V из списка вершин, **конец**.

В приведенном алгоритме порядок порождения новых вершин таков, что при выборе очередной вершины для ветвления предпочтение будет отдаваться вершинам с более равномерным размещением работ по вычислительным модулям.

Докажем два утверждения.

Утверждение 1. Оценка снизу для любой вершины не меньше, чем оценка снизу для родительской вершины.

Доказательство. Рассмотрим вершину V и порожденную от неё вершину V'. $H^{V'}$, наиболее полное расписание для V', не может содержать меньше работ и сообщений, чем расписание H^V. Т.к. иначе в качестве наиболее полного расписания для V' можно было бы взять H^V.

Следствие 1. Оценка снизу равна гарантированному (минимальному) количеству работ и сообщений, которое может быть запланированно в наиболее полном корректном расписание для любой вершины в поддереве с корнем в данной вершине.

Утверждение 2. Оценка сверху для любой вершины не больше, чем оценка сверху для родительской вершины.

Доказательство. Рассмотрим вершину V и порожденную от неё вершину V'. Для доказательства утверждения достаточно доказать, что количество работ и сообщений не попавших в $H^{V'}$ не может быть меньше того же количества для H^v. Пусть в H^V были запланированы все работы из W^V и все сообщения из M^V, а в V' добавляется новая работа i, т.е. $W^{V'} = W^V \cup \{i\}$. Возможны следующие варианты для планирования работы i:

1) запланирована вместе со всеми работами из W^V;

2) не запланирована;

3) запланирована, но одна из работ из W^V оказалась незапланированной.

В первом варианте оценка сверху не измениться. Во втором и третьем уменьшится на единицу, т.к. количество незапланированных работ увеличилось на одну. Таким образом, оценка сверху для V' будет не больше, чем оценка для V. Аналогичным образом рассматриваются случаи, когда в H^V запланированы не все работы из W^V.

Следствие 2. Оценка сверху равна максимально возможному количеству работ и сообщений, которое может быть запланированно в наиболее полном корректном

расписание для любой вершины в поддереве с корнем в данной вершине.

3.3. Алгоритм построения расписания при заданном размещении работ

Алгоритм, рассматриваемый в данном разделе, используется для построения расписания для вершин дерева решений основного алгоритма. Для ясности будем называть описываемый ниже алгоритм внутренним. На вход внутреннего алгоритма поступают данные вершины V основного алгоритма, а именно — размещение A^V набора работ W^V и набор сообщений M^V. Он также основан на методе ветвей и границ и осуществляет для каждого вычислительного модуля упорядочивание работ, размещенных на нем, а для среды передачи данных — упорядочивание сообщений. Каждой вершине VV дерева решений внутреннего алгоритма соответствуют упорядоченные списки работ $W_{p_l}^{VV} \subseteq W^V$, по одному на каждый вычислительный модуль: $W_{p_l}^{VV} = \left\{j \in W^V : (j, p_l) \in A^V\right\}$, и список сообщений $M^{VV} \subseteq M^V$. В соответствии с порядком, задаваемым этими списками, строится совместное расписание H^{VV} для вершины VV. Алгоритм построения расписания по заданному порядку работ и сообщений рассмотрен в следующем разделе. Начальной вершине соответствует набор пустых списков. Аналогично основному алгоритму для ветвления выбирается вершина согласно оценкам сверху и снизу, при равенстве оценок приоритет отдаётся вершине, порожденной раньше.

Алгоритм 2 (Процедура ветвления внутреннего алгоритма).

1. Каждому вычислительному модулю p_l сопоставить множество $W_{p_l}^{cands}$ незапланированных в H^{VV} работ, размещенных согласно A^V на этот модуль, и не имеющих незапланированных в H^{VV} предшественников.

2. Среде передачи данных сопоставить множество $M_{p_l}^{cands}$ незапланированных в H^{VV} сообщений, не имеющих незапланированных в H^{VV} предшественников.

3. Отсортировать построенные множества согласно критерию сравнения $<_{H^{VV}}$ и пронумеровать их элементы.

4. Построить все возможные варианта выбора по одному элементу из каждого составленного на шагах 1 и 2 множеств.

5. Для каждого из построенного варианта выбора, когда отобраны элементы $j_1, \cdots, j_{n_P} \in W^V, k \in M^V$, породить новую вершину VV'. Вершина VV' отличается от вершины VV тем, что в конец каждого из списков $W_{p_l}^{VV}$ добавлена работа j_l, а в конец списка M^{VV} — сообщение k. Для вариантов выбора, имеющих меньший лексикографический номер вершина порождается раньше, чем для вариантов, имеющих больший лексикографический номер.

Таким образом, при порождении новой вершины в конец каждого из списков по возможности добавляется новый элемент.

После построения расписания для вершины VV осуществляется её оценка сверху и снизу аналогично оценке вершин основного алгоритма. Для более раннего отсечения ветвей, не содержащих оптимального решения, оценка сверху дополнительно уточняется. Для работ и сообщений, которые не находятся в списках $W_{p_l}^{VV}$ и M^{VV}, т.е. из множеств $W^V \setminus W^{VV}$ и $M^V \setminus M^{VV}$ при помощи полиномиального алгоритма, описанного в [6], строятся одноприборные расписания с прерываниями без учета технологических ограничений и ограничений на порядок работ и сообщений. Оценка сверху уменьшается на число работ и сообщений, запланированных в этих расписаниях с нарушением директивных сроков.

Для нижней и верхней оценок вершин внутреннего алгоритма могут быть доказаны аналоги утверждений 1 и 2 для вершин основного алгоритма.

3.4. Алгоритм построения расписания по заданному порядку работ и сообщений

Рассматриваемые в данной работе технологические ограничения на расписание передачи сообщений предъявляют жесткие требования к паузам между сообщениями. Это приводит к тому, что в некоторых случаях для построения оптимального расписания выгодно планировать передачу сообщений на самое раннее возможное время, а позже. Проиллюстрируем это на примере.

Пример 1. Множество сообщений состоит из трех сообщений с номерами 1, 2 и 3. Характеристики сообщений приведены в таблице:

Сообщение	s_i	f_i	dm_i
1	0	10	1
2	0	10	2
3	5	9	3

Технологические ограничения следующие: максимальная длительность цепочки сообщений $r_{mct} = 6$, максимальное количество сообщений в цепочке $r_{mcc} = 3$, минимальный интервал между цепочками $r_{mi} = 4$. Сообщения должны передаваться в таком же порядке, т.е. сначала 1, потом 2, затем 3.

Если запланировать начало передачи сообщения 1 как только наступит её директивный срок начала передачи, а сообщение 2 — как только закончится передача сообщения 1, то сообщение 3 не сможет быть передано в рамках своего директивного интервала (см. рис. 3).

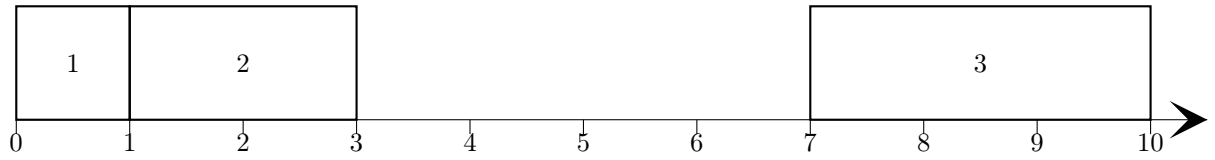

Рис. 3. Сообщение 3 не укладывается в свой директивный интервал

Если же запланировать начало передачи сообщения 1 на более позднее время, то сообщение 3 будет передаваться в рамках своего директивного интервала (см. рис. 4).

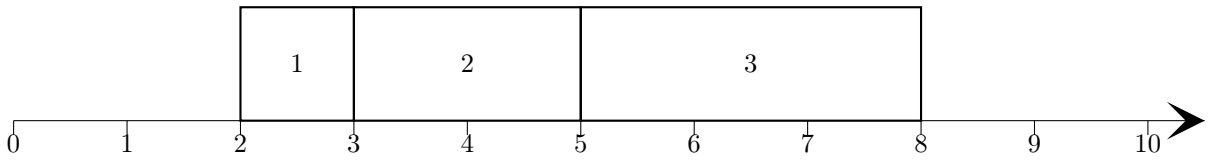

Рис. 4. Сообщение 3 укладывается в свой директивный интервал

Предлагаемый алгоритм сначала пробует запланировать все работы и сообщения (из множеств W^{VV} и M^{VV}) на самое раннее возможное время (которое вычисляется исходя из директивного интервала, времени завершения предшественников и занятости устройств). Если какие-то работы не укладываются в свои директивные интервалы, то расписание строится заново, но уже без них и их потомков. Если какое-то сообщение не

укладывается в свой директивный интервал, то строятся два расписания: расписание без этого сообщения и его потомков и расписание, в котором цепочка сообщений перед этим сообщением сдвинута на такое время, что сообщение попадает в конец цепочки и укладывается в свой директивный интервал. Затем из двух расписаний выбирается то, в котором больше число запланированных работ и сообщений.

Входными данными алгоритма построения расписания по заданному порядку работ и сообщений являются упорядоченные списки работ W_p^{VV}, по одному на каждый вычислительный модуль, и упорядоченный список сообщений M^{VV}, соответствующие какой-либо вершине VV внутреннего алгоритма. Результатом работы алгоритма является корректное расписание, содержащее максимально возможное количество элементов W_p^{VV} и M^{VV} и в котором порядок выполнения работ и передачи сообщений соответствует порядку, задаваемому этими списками.

Опишем вспомогательную процедуру *move* принимающую на вход упорядоченные списки работ $\widetilde{W}_p^{VV}$ и сообщений $\widetilde{M}^{VV}$, полученные удалением некоторых элементов из списков W_p^{VV} и M^{VV}, и множество заданий S, в котором каждому заданию i сопоставлено время s_i, раньше которого нельзя планировать начало выполнения или передачи этого задания. Результатом работы процедуры является расписание H, возможно некорректное.

Алгоритм 3 (Процедура *move*).

1. Запланировать в расписание H каждую работу из списков $\widetilde{W}_p^{VV}$ и каждое сообщение из списка $\widetilde{M}^{VV}$ в начало своего директивного интервала.
2. Выбрать произвольный вычислительный модуль p.
3. Выбрать первую работу i из списка $\widetilde{W}_p^{VV}$.
4. Вычислить минимально возможное время c_i для планирования работы i, считая запланированными на p только те работы, которые в списке идут раньше, чем i.
5. Если работа i содержится в множестве S, то скорректировать время $c_i = \max(c_i, s_i)$.
6. Если $c_i > \hat{s}_i^w$, то перепланировать работу i на время c_i.
7. Если не все работы из списка $\widetilde{W}_p^{VV}$ были обработаны, то выбрать в качестве i следующую работу из списка и перейти на шаг 4.
8. Если ещё не все вычислительные модули были обработаны, то выбрать в качестве p другой ещё необработанный модуль и перейти на шаг 3.
9. Выбрать первое сообщение i из списка $\widetilde{M}^{VV}$.
10. Вычислить минимально возможное время c_i для планирования сообщения i, считая запланированными только те сообщения, которые в списке идут раньше, чем i.
11. Если сообщение i содержится в множестве S, то скорректировать время $c_i = \max(c_i, s_i)$.
12. Если $c_i > \hat{s}_i^m$, то перепланировать сообщение i на время c_i.
13. Если не все сообщения из списка $\widetilde{M}^{VV}$ были обработаны, то выбрать в качестве i следующее сообщение из списка и перейти на шаг 10.
14. Если на шаге 6 или на шаге 12 были перепланированы какие-либо работы или сообщения, то перейти на шаг 2, иначе **конец**.

Процедура *move* работает таким образом, что корректность результирующего расписания может нарушаться только за счет нарушения директивных сроков завершения каких-либо работ или сообщений.

Теперь опишем рекурсивную процедуру *process*, входные данные которой совпадают с входными данными процедуры *move*, а результатом работы является корректное расписание H.

Алгоритм 4 (Процедура *process*).

1. $H = move\left(\widetilde{W}_1^{VV}, \cdots, \widetilde{W}_{n_P}^{VV}, \widetilde{M}^{VV}, S\right)$.
2. Если H является корректным расписанием, то **конец**.
3. Найти самое раннее задание i, чей директивный срок завершения нарушается.
4. Если i является работой, то $H = process\left(\widetilde{W}_1^{VV}, \cdots, \widetilde{W}_p^{VV}|i, \cdots, \widetilde{W}_{n_P}^{VV}, \widetilde{M}^{VV}, S\right)$, где $\widetilde{W}_p^{VV}|i$ — это список $\widetilde{W}_p^{VV}$, из которого удалена работа i и все её потомки, **конец**.
5. Если i является сообщением, то пусть l – первое сообщение цепочки сообщений, которой принадлежит i, а j и k — первое и последнее сообщение предыдущей цепочки.
6. Построить два расписания: $H_1 = process\left(\widetilde{W}_1^{VV}, \cdots, \widetilde{W}_{n_P}^{VV}, \widetilde{M}^{VV}|i, S\right)$ и $H_2 = process\left(\widetilde{W}_1^{VV}, \cdots, \widetilde{W}_{n_P}^{VV}, \widetilde{M}^{VV}, \widetilde{S}\right)$, где $\widetilde{S}$ — это множество S, в которое добавлено сообщение j с временем $s_j = s_l^m - \hat{s}_k^m - dm_k$.
7. Если $|H_1| > |H_2|$, то $H = H_1$, иначе $H = H_2$, **конец**.

Если сообщение не укладывается в свой директивный интервал, то в процедуре *process* строятся и сравниваются два расписания: расписание без этого сообщения и его потомков и расписание, в котором это сообщение принадлежит другой цепочке сообщений (см. пример). Из двух расписаний выбирается лучшее.

Алгоритм построения расписания по заданному порядку работ и сообщений состоит в вызове процедуры *process* со следующими параметрами: списками, приписанными вершине VV и пустым множеством S, т.е. $process\left(W_1^{VV}, \cdots, W_{n_P}^{VV}, M^{VV}, \varnothing\right)$.

4. Экспериментальное исследование построенного алгоритма

Для исследования алгоритма были случайным образом составлены девять наборов исходных данных, для которых существует полное и корректное расписание. В состав всех наборов входит 300 работ, 100 сообщений и 4 вычислительных модуля одинаковой производительности. Длительность выполнения каждой работы лежит на отрезке от 500 до 1500 мс. Количество слов данных в каждом сообщении лежит на отрезке от 1 до 32 (длительность передачи - от 104 до 724 мс). Длительность интервала планирования равна 500000 мс. Характеристики среды обмена следующие: $r_{mct} = 500$ мс, $r_{mcc} = 10$ и $r_{mi} = 500$ мс. Для всех построенных наборов исходных данных алгоритмом были найдены полные и корректные расписания. Среднее время работы алгоритма на компьютере с процессором Intel Core i5-2310 с тактовой частотой 2.9 ГГц составило чуть более 3 часов.

5. Заключение

В данной статье рассматривалась задача совместного планирования вычислений и обменов в информационно-управляющих системах реального времени (ИУС РВ). Данная задача возникает при модификации существующих ИУС РВ или разработке новых на базе существующих. Была описана её формальная постановка. Для решения задачи был предложен алгоритм на основе метода ветвей и границ, который гарантирует нахождение точного решения, т.е. наиболее полного и корректного расписания.

СПИСОК ЛИТЕРАТУРЫ

1. *Костенко В.А., Шестов П.Е.* Жадный алгоритм совместного планирования вычислений и обменов в системах реального времени // Известия РАН. Теория и системы управления. 2012. № 5. С. 35–49.

2. *Шестов П.Е., Костенко В.А.* Анализ подходов к совместному планированию вычислений и обменов в системах реального времени // Программные системы и инструменты. Тематический сборник. 2011. № 12. С. 172–182.

3. *Peng D.-T., Shin K. G., Abdelzaher T. F.* Assignment and scheduling of communicating periodic tasks in distributed real-time systems // IEEE Transactions on Software Engineering, 1997. V. 23. No. 12. P. 745–758.

4. *Cheng S., Agrawala A.* Allocation and scheduling of real-time periodic tasks with realative timing constraints // In Proc. of the Second International Workshop on Real-Time Computing Systems and Applications, 1995. P. 210–217.

5. *Сухарев А. Г., Тимохов А. В., Федоров В. В.* Курс методов оптимизации. М.: ФИЗМАТЛИТ, 2005. 368 с.

6. *K.R. Baker et al.* Preemptive Scheduling of a Single Machine to Minimize Maximum Cost Subject to Release Dates and Precedence Constraints // Operations Research 1983, V. 31, No. 2, P. 381–386