Костенко В.А. Задача построения расписания при совместном проектировании аппаратных и программных средств // Программирование - 2002. - №3 - С.64-80. **(Kostenko V.A. The Problem of Schedule Construction in the Joint Design of Hardware and Software. // Programming and Computer Software – 2002. -Vol. 28 - № 3 - pp.162–173.).**

ЗАДАЧА ПОСТРОЕНИЯ РАСПИСАНИЯ ПРИ СОВМЕСТНОМ ПРОЕКТИРОВАНИИ АППАРАТНЫХ И ПРОГРАММНЫХ СРЕДСТВ.

Костенко В.А.
Факультет вычислительной математики и кибернетики МГУ
119899, Москва, Воробьевы горы

В работе рассмотрены структуры данных для непосредственного и параметрического представления расписаний и введены соответствующие системы операций коррекции расписаний для построения итерационных алгоритмов. Доказано, что непосредственный и параметрический способы представления расписаний и соответствующие операции коррекции, допускают построение итерационных алгоритмов, обеспечивающих переход от произвольного допустимого расписания к оптимальному за линейное число итераций относительно числа планируемых работ и исключают возможность получения недопустимых расписаний на всех итерациях алгоритма.

Введение

Можно выделить два базовых варианта задачи совместного проектирования аппаратных и программных средств вычислительных систем (ВС) реального времени:

1) минимизируется время выполнения прикладной программы (расписания выполнения программы) при заданных ограничениях на аппаратные ресурсы (ВС мягкого реального времени);
2) минимизируются аппаратные ресурсы при ограничении времени выполнения прикладной программы (ВС жесткого реального времени).

Частные варианты задачи получаются из базовых, выбором набора варьируемых характеристик аппаратных средств и выбором функции для оценки затрат аппаратных средств. В качестве меры эффективности расписания могут быть использованы и другие критерии: среднее взвешенное время завершения, время ожидания обслуживания, среднее количество процессов, выполненное на заданном временном интервале.

Основной особенностью задачи построения расписания при совместном проектировании аппаратных и программных средств ВС является то, что расписание строится одновременно с синтезом архитектуры (определением оптимизируемых параметров архитектуры). Эффективность и применимость методов построения расписания при совместном проектировании аппаратных и программных средств, кроме точности и вычислительной сложности методов, также определяется:

- универсальностью методов построения расписания в классе допустимых архитектур ВС, которые могут быть получены в результате синтеза;
- совместимостью методов построения расписания и методов выбора оптимальных параметров архитектуры;
- возможностью методов построения расписания строить расписания для архитектур ВС с различным уровнем детализации.

В первом разделе рассматривается понятие инварианта поведения программы, во втором разделе расписание рассмотрено как комбинаторная структура и введены ограничения на расписание, при которых сохраняется инвариант поведения программы. В третьем разделе проведен анализ различных подходов к разработке алгоритмов построения расписания при совместном проектировании аппаратных и программных средств ВС. В четвертом и пятом разделах рассматриваются непосредственные и параметрические способы

представления расписания и вводятся функционально полные системы операций коррекции расписаний.

1. Модель прикладной программы

При определении модели прикладной программы предполагаем, что выделение работ, подлежащих планированию, и параллелизм, допускаемый при выполнении программы заданы (выявлены) предварительно.

В качестве модели прикладной программы будем использовать частный случай инварианта поведения программы предложенного в работах [1,2]. Следуя [1], обозначим поведение программы *Bh(PR)* и определим *Bh(PR)* следующим образом: $Bh(PR) = < S, \{R — (PR)\}, \{R \rightarrow (PR)\}>$, где S – множество всех возможных шагов процессов в допустимом диапазоне входных данных программы, $\{R \rightarrow (PR)\}$ - отношения, определяющие частичный порядок на множестве шагов каждого процесса, $\{R — (PR)\}$ - отношения взаимодействия между процессами.

Шаги процесса определяются последовательностью взаимодействий с другими процессами [1]. Назовем рабочим интервалом процесса внутренние действия процесса между двумя последовательными взаимодействиями с другими процессами. Каждый рабочий интервал процесса по существу является реализацией соответствующего шага процесса.

Для задачи синтеза архитектур будем использовать одну из историй поведения программы $H(PR) \in Bh(PR)$ [1] (поведение программы для конкретного набора входных данных). Для *H(PR)* отношение $\{R \rightarrow (PR)\}$ является отношением полного порядка, а множество S сужается до множества шагов, которые делают процессы для конкретного набора входных данных.

H(PR) можно представить ациклическим ориентированным размеченным графом. Вершинам $P = \{p_i\}_{i=1}^{N_1} \cup \{p_i\}_{i=1}^{N_2} \cup \ldots \cup \{p_i\}_{i=1}^{N_K}$, соответствуют рабочие интервалы процессов, дугам $\prec = \{\prec_{ik} = (p_i, p_k)\}_{(i,k) \in (1 \ldots N)}$ - связи, определяющие взаимодействия между рабочими интервалами из множества P (определяются объединением отношений $\{R—(PR)\}, \{R \rightarrow (PR)\}$). Где N_i - число рабочих интервалов в i-ом процессе, K - число процессов в программе *PR*, $N = N_1 + N_2 + \ldots + N_K$ - мощность множества P. Чередование рабочих интервалов различных процессов, назначенных на один и тот же процессор, допустимо, если не нарушается частичный порядок, заданный $\prec$. Отношение $\prec_{ik}$ представляется следующим образом: если $p_i \prec_{ik} p_k$, то рабочий интервал p_i, необходимо выполнить до начала выполнения рабочего интервала p_k. На $\prec$ накладываются условия ацикличности и транзитивности. Каждая вершина имеет свой уникальный номер и метки: принадлежности рабочего интервала к процессу и вычислительной сложности рабочего интервала. Вычислительная сложность рабочего интервала позволяет оценить время выполнения рабочего интервала на процессоре. Дуга определяется номерами смежных вершин и имеет метку, соответствующую объему данных обмена. Объем данных обмена для каждой связи из $\prec$ позволяет оценить затраты времени на выполнение внешнего взаимодействия.

При сделанных выше допущениях используемый инвариант поведения программы будет частным случаем (одна история) инварианта поведения программы, определенном в [2]. Используемый инвариант поведения определим:

1. Множеством рабочих интервалов процессов, составляющих программу *PR*:

   $P = \{p_i\}_{i=1}^{N_1} \cup \{p_i\}_{i=1}^{N_2} \cup \ldots \cup \{p_i\}_{i=1}^{N_K}$,

   Нумерация рабочих интервалов является сквозной и удовлетворяет условиям полной топологической сортировки. Каждый рабочий интервал имеет метку принадлежности к процессу.
2. Частичным порядком на P:

$\prec = \{\prec_{ik} = (p_i, p_k)\}_{(i,k) \in (1...N)}$;

3. Вычислительной сложностью рабочих интервалов:

$\{w_i\}_{i=1}^{N}$;

4. Объемом данных обмена для каждой связи из $\prec$:

$\{v_{ik}\}_{(i,k) \in (1...N)}$;.

2. Расписание

Расписание выполнения программы определено, если заданы: 1)множества процессоров и рабочих интервалов, 2)привязка, 3)порядок. Привязка - всюду определенная на множестве рабочих интервалов функция, которая задает распределение рабочих интервалов по процессорам. Порядок задает ограничения на последовательность выполнения рабочих интервалов и является отношением частичного порядка, удовлетворяющим условиям ацикличности и транзитивности. Отношение порядка на множестве, рабочих интервалов распределенных на один и тот же процессор, является отношением полного порядка.

Модель расписания выполнения программы определим набором простых цепей и отношением частичного порядка $\prec_{HP}$ на множестве P: $HP=(\{SP_i\}_{i=(1...M)}, \prec_{HP})$.
Где $\{SP_i\}_{i=(1...M)}$ -набор простых цепей (ветвей параллельной программы). Они образуются рабочими интервалами процессов, распределенными на один и тот же процессор (M – число процессоров в ВС). Отношение частичного порядка $\prec_{HP}$ на множестве P для HP определим как объединение отношений: $\prec_{HP} = \prec_c \cup \prec_1 \cup \ldots \cup \prec_M$, $\prec_i$ - отношение полного порядка на SP_i, которое определяется порядковыми номерами рабочих интервалов в SP_i; $\prec_c$ - набор секущих ребер, которые определяются связями рабочих интервалов, распределенных на разные процессоры. Если рабочие интервалы p_i и p_j распределены на разные процессоры и в $\prec$ существует связь $\prec_{ij}$, то она определяет секущее ребро в HP. На отношение $\prec_{HP}$ накладываются условия ацикличности и транзитивности.

Модель расписания можно рассматривать как граф HP, вершины которого имеют дополнительную метку “номер списка”, а дуги определяются отношением $\prec_{HP}$ или как граф H, вершины которого доразмечены двойками: {“номер списка”, “порядковый номер вершины в соответствующем списке”}. В модели HP сохраняются нумерация вершин, дуг и их метки заданные в модели поведения программы H. В дальнейшем при рассмотрении свойств расписаний в некоторых разделах будет использоваться ярусная форма представления расписания [3]. Ярусной формой максимальной высоты будем называть такую ярусную форму, у которой на каждом ярусе находится не более одной вершины.

Ограничения на расписание

Расписание HP является допустимым (сохраняет инвариант поведения программы), если выполнены следующие ограничения:

1. Каждый рабочий интервал должен быть назначен на процессор (в SP_i).
2. Каждый рабочий интервал должен быть назначен лишь на один процессор (в один SP_i).
3. Частичный порядок, заданный в H сохранен в HP:
   $\prec \subset \prec_{HP}^{T}$, $\prec_{HP}^{T}$ – *транзитивное замыкание отношения* $\prec_{HP}$.
4. Расписание HP должно быть беступиковым. Условием беступиковости является отсутствие контуров в графе HP: $\prec_{HP}$ – *ациклично*.
5. Все рабочие интервалы одного процесса должны быть назначены на один и тот же процессор (в один и тот же SP_i).

Ограничения 1-4 обеспечивают сохранение инварианта поведения программы и являются обязательными. Ограничение 5 запрещает возобновление работы процесса после прерывания на другом процессоре, т.е. определяет способ организации параллельных вычислений в ВС, и не всегда является обязательным. В дальнейшем будем говорить, что расписание допустимо $HP \in HP^{*}_{1-5}$, если оно удовлетворяет ограничениям 1-5. Нижний индекс в HP^{*}_{1-5} указывает ограничения налагаемые на расписание.

3. Анализ различных подходов для решения задачи построения расписаний

Понятия динамических критериев и ограничений

Значения динамических критериев и ограничений для конкретного варианта решения задачи синтеза архитектур ВС могут быть получены из анализа динамики функционирования ВС при выполнении прикладной программы. Для описания динамики функционирования ВС выделен набор состояний рабочих интервалов и аппаратных ресурсов (приложение 1). Динамику функционирования ВС *TR* будем определять множеством событий привязанных к временной оси, происходящих при выполнении расписания *HP*, на архитектуре *HW*. Под событием будем понимать переход любого аппаратного ресурса и рабочего интервала из одного состояния в другое.

Динамика функционирования ВС может быть получена совместной интерпретацией моделей *HW* и *HP*: *TR = φ(HW, HP)*. Можно выделить следующие основные уровни интерпретации, определяемые набором временных задержек, учитываемых динамически (т.е. величина задержки определяется состоянием системы в конкретный момент времени поступления запроса на ресурс или данные) при вычислении функции *TR = φ(HW, HP)*:

1. $t^{EP}, t^{D\prec}$,
2. $t^{EP}, t^{D\prec}, t^{EAS}$,
3. $t^{EP}, t^{D\prec}, t^{EAS}, t^{SYS}$,

где t^{EP}- временные задержки, обусловленные занятостью процессоров, $t^{D\prec}$- ожиданием завершения выполнения предшественников, t^{EAS}- ожиданием получения доступа к разделяемым ресурсам и t^{SYS}- задержки вносимые логической средой и доступом к памяти, которые не могут быть учтены статически.

Для первых двух уровней интерпретации оценка времени выполнения рабочего интервала (время нахождения в состоянии *SP_W*) по его вычислительной сложности не зависит от расположения рабочего интервала в расписании и может быть получена статически. На первом уровне интерпретации вычислительную сложность рабочих интервалов можно задавать временем выполнения. С учетом отсутствия задержек на получение разделяемых ресурсов в это время можно включить затраты времени на взаимодействие с рабочими интервалами других процессов. На втором уровне интерпретации время нахождения рабочего интервала в состоянии *SP_EAS* не может быть определено статически.Для третьего уровня интерпретации оценка времени нахождения рабочего интервала в состоянии *SP_SPR* (ожидание обновления данных в памяти процессора или завершения, вызванного системного процесса) может потребовать знания семантики используемых данных и соответственно может зависить от расположения рабочего интервала в расписании.

Набор динамически учитываемых временных задержек определяет точность вычисления оценок значений динамических критериев и ограничений для различных классов архитектур и чем выше уровень интерпретации, тем выше требуемый уровень детализации представления архитектуры ВС. Например, для полносвязной архитектуры достаточно динамически учитывать лишь задержки $t^{EP}, t^{D\prec}$, а для архитектур с коммутационной средой, в которой возможно ожидание получения канала связи, динамический учет лишь $t^{EP}, t^{D\prec}$ и статический t^{EAS} будет приводить к неточным оценкам.

В соответствии с данными уровнями интерпретации будем выделять следующие уровни проектирования: 1)абстрактный, 2)системный и 3)уровень регистровых передач.

Независимо от уровня интерпретации, оператор интерпретации всегда должен удовлетворять условию: $HP = \varphi^{-1}(\varphi(HW, HP))$, т.е. расписание однозначно восстанавливается по динамике функционирования ВС. Расписание *HP* интерпретируемо, если отношение порядка ациклично и транзитивно, а привязка - всюду определеная функция на множестве процессов. Для выполнения оператора интерпретации, независимо от конкретного варианта постановки задачи синтеза архитектур ВС, всегда должно быть построено расписание выполнения программы *HP*. Что, собственно говоря, и обеспечивает важность параметра и его присутствие в большинстве вариантов постановки задачи синтеза архитектур.

Задача построения расписания при совместном проектировании аппаратных и программных средств, классы алгоритмов для ее решения

При разработке ВС реального времени можно выделить два базовых варианта задачи совместного проектирования аппаратных и программных средств:

3) минимизируется время *T* выполнения расписания при заданных ограничениях на аппаратные ресурсы (ВС мягкого реального времени);
4) минимизируются аппаратные ресурсы при ограничении времени выполнения расписания $T \le T^{dir}$ (ВС жесткого реального времени).

Частные варианты задачи получаются из базовых, выбором набора варьируемых характеристик аппаратных средств и выбором функции для оценки затрат аппаратных средств. В качестве меры эффективности расписания могут быть использованы и другие критерии [4]: среднее взвешенное время завершения, время ожидания обслуживания, среднее количество процессов, выполненное на заданном временном интервале.

В дальнейшем в качестве динамического критерия для определенности будем рассматривать время выполнения расписания: *T=f(HP,HW)*. Функция *T=f(HP,HW)* задана алгоритмически: правилами/алгоритмами ее вычисления. Выбор наиболее подходящего метода решения определяется как подклассом задачи совместного проектирования аппаратных и программных средств ВС, так и характеристиками функции *T=f(HP,HW)*. Функцию *T=f(HP,HW)* будем характеризовать:

1) размером моделей *HP,HW*:
   - *HP*: число рабочих интервалов (*N*), число процессов (*K*), мощность исходного отношения порядка ($/ \prec /$),
   - *HW*: число процессоров (*M*), число разделяемых ресурсов (M^{EAS}), число уровней памяти (M^{MEM}), число вызываемых системных процессов (M^{SYS});
2) структурой модели *H*:
   - тип $\prec$,
   - дисперсия $\{w_i\}_{i=1}^{N}$,
   - суммарная вычислительная сложность рабочих интервалов и критический путь;
3) набором временных задержек, учитываемых динамически (см. уровни интерпретации):
   - $t^{EP}, t^{D\prec}$,
   - $t^{EP}, t^{D\prec}, t^{EAS}$,
   - $t^{EP}, t^{D\prec}, t^{EAS}, t^{SYS}$;
4) законом распределения значений *T*: $\rho_i=\rho(T_i)$. Значение функции $\rho(T_i)$ равно относительному числу допустимых вариантов расписания для которых значение функции *T=f(HP,HW)* равно T_i.

В работах [4,5] показано, что почти все задачи построения расписаний являются *NP*-полными в строгом смысле. В данной работе на вариантах постановки задач построения расписаний, для которых существуют полиномиальные алгоритмы, останавливаться не будем, достаточно полный обзор таких задач и алгоритмов приведен в [4]. Отметим, что

большинство таких задач являются практически вырожденными, поскольку на исходные условия накладываются жесткие ограничения: тип исходного отношения – лес или пустое, число процессоров менее трех, одинаковая вычислительная сложность рабочих интервалов, дополнительные ресурсы не допустимы, допустимость прерывания выполнения любого рабочего интервала и возобновление его выполнения на любом из процессоров ВС. Данным ограничениям большинство реальных ВС и программ не удовлетворяют.

Анализ различных подходов к разработке алгоритмов построения расписаний

Возможный широкий спектр частных вариантов задачи и характеристик функции *T=f(HP,HW)* обуславливает широкий спектр используемых алгоритмов. Алгоритмы построения расписаний можно разбить на три большие группы:

- алгоритмы, основанные на декомпозиции задачи составления расписаний на подзадачи (вложении задачи в семейство более простых задач);
- алгоритмы, основанные на методе ветвей и границ;
- алгоритмы, основанные на коррекции текущего расписания (итерационные алгоритмы).

Алгоритмы, использующие декомпозицию расписаний, могут быть основаны:

- на динамическом программировании [6,7];
- на жадных стратегиях [8,9,10].

Алгоритмы, основанные на динамическом программировании, требуют введения двух операций: 1)вложение решаемой задачи в семейство более простых задач; 2)нахождение возвратного соотношения, связывающего оптимальные значения этих подзадач. Данные алгоритмы позволяют получить глобальный оптимум, но при этом для большинства задач комбинаторной оптимизации имеют не полиномиальную сложность. В частности, для *NP*-полных задач составления расписаний сложность алгоритмов – факториальная функция от размера входа (число процессов, число рабочих интервалов и мощность множества исходного отношения частичного порядка).

Какие-либо жадные стратегии, гарантированно получающие оптимальное расписание, на настоящий момент времени не известны, за исключением небольшого числа вариантов задач составления расписаний, которые не принадлежат к классу *NP*-полных. Все жадные эвристические алгоритмы подразумевают существование критериев оптимизации для решения подзадач (локальных целевых функций). Решение о распределении очередного процесса принимается исходя из принципа получения оптимального расписания на текущем шаге при условии, что расписание, полученное на предыдущем шаге, не может быть изменено. То есть, на каждом шаге алгоритм делает выбор, оптимальный с точки зрения получения частичных расписаний, предполагая, что эти локально-оптимальные решения приведут к приемлемому решению задачи.

Применение жадных алгоритмов для составления расписаний при совместном проектировании аппаратных и программных средств ограничивается [11-14]:

- классом архитектур без последействия или даже без разделяемых ресурсов, если их влияние на значение функции *T=f(HP,HW)* не может быть локализовано;
- проблемой выбора локальных целевых функций индивидуально для каждой подзадачи;
- жадные алгоритмы построения расписания совместимы с алгоритмами выбора оптимальных параметров архитектуры в рамках метода покоординатного спуска.

Основной областью эффективного применения жадных алгоритмов при решении задачи совместного проектирования аппаратных и программных средств ВС может быть решение задачи “оценка принципиальной возможности построения ВС при заданных ограничениях и сужение класса рассматриваемых архитектур” на абстрактном уровне проектирования ВС. При решении этой задачи структура ВС может рассматриваться как полносвязная и достаточен лишь первый уровень интерпретации. Определяется число

процессоров в ВС, строится расписание выполнения программы, определяются требования к коммутационной среде и памяти, и проводится оценка возможности различных способов их реализации при заданных ограничениях.

Алгоритмы, основанные на методе ветвей и границ [4,7], используют разделение пространства возможных решений (представление в форме некоторого разветвления), верхнюю/нижнюю оценку значения целевой функции для выделенных областей и отсечение областей в которых нет оптимального решения. Данные алгоритмы позволяют получить глобальный оптимум, но при этом для большинства задач комбинаторной оптимизации имеют не полиномиальную сложность. В частности, для *NP*-полных задач составления расписаний сложность алгоритмов – факториальная функция от размера входа.

Алгоритмы, основанные на коррекции текущего расписания, (итерационные алгоритмы) можно разбить на два подкласса:

- алгоритмы, опирающиеся на метод проб и ошибок: генетические и эволюционные алгоритмы [15-18], алгоритмы имитации отжига [19,20], алгоритмы случайного поиска (ненаправленного, направленного, направленного с самообучением)[20,21];
- алгоритмы детерминированной коррекции расписания.

Схематично работу итерационных алгоритмов для решения задачи построения расписания можно представить следующим образом:

1. Задать начальное приближение HP^0, $k=0$.
2. Вычислить целевую функцию $f(HP^k,HW)$ и проверить выполнение ограничений (если в п.3 возможно получение недопустимых значений HP).
3. Получить $HP^{k+1}=D(\{HP^i\},\{f(HP^i,HW)\},Pr^k:i\in(1,\ldots,k))$.
4. Если заданный критерий останова не достигнут, то $k=k+1$ и перейти к п.2; в противном - завершить работу алгоритма.

Где, D – некоторая стратегия коррекции текущего расписания, Pr^k – параметры стратегии (для ряда стратегий, возможно, их изменение в ходе работы алгоритма).

Алгоритмы, опирающиеся на метод проб и ошибок, содержат элементы случайного выбора привязки и порядка, что дает возможность получать информацию о влиянии привязки и порядка на значения целевой функции и ограничений. Данная информация используется для адаптации алгоритма (в ходе его работы) к закону влияния привязки и порядка на значения целевой функции и ограничений. Кроме возможности адаптации к решаемой задаче, алгоритмы, опирающиеся на метод проб и ошибок, наиболее полно удовлетворяют требованиям, к свойствам алгоритмов построения расписания при совместном проектировании аппаратных и программных средств ВС: 1)возможностям методов строить расписания для архитектур ВС с различным уровнем детализации; 2)универсальности методов в классе допустимых архитектур ВС; 3)совместимости методов построения расписания и методов выбора оптимальных параметров архитектуры.

Данные особенности алгоритмов обеспечивают эффективность их использования на системном уровне проектирования: определение числа процессоров, типа каждого процессора (тип процессора характеризуется стоимостью и производительностью), топологии и характеристик компонентов коммутационной среды, распределение памяти (тип – ПЗУ/ОЗУ, объем, способ доступа – локальная/разделяемая, циклы считывания/записи) и построение расписания выполнения прикладной программы.

Алгоритмы детерминированной коррекции расписания для своего построения требуют некоторых априорных сведений о влиянии привязки и порядка на значения целевой функции и ограничений. Это обуславливает специализацию методов на конкретный класс архитектур ВС. Однако, для уточнения расписания (локальной оптимизации расписания) можно предложить алгоритмы детерминированной коррекции расписания исправляющие “плохие” фрагменты расписания и не использующие априорные сведения о влиянии привязки и порядка на значения целевой функции и ограничений. Если некоторый рабочий

интервал находится длительное время в состояниях *SP_D* - ожидание данных от других рабочих интервалов или *SP_EAS* - ожидание получения разделяемого ресурса, то можно попытаться уменьшить время нахождения рабочего интервала в этих состояниях путем изменения расположения в расписании рабочего интервала или его непосредственных предшественников. То есть, работа алгоритма основана на использование информации о “плохих” фрагментах расписания, которая может быть определена при анализе динамики функционирования ВС (*TR)*. Такое построение алгоритмов детерминированной коррекции расписания позволяет устранить их узкую специализацию на конкретный класс архитектур ВС. Основная область эффективного применения алгоритмов детерминированной коррекции решений – уточнение расписаний на уровне регистровых передач при решении задачи уточнения деталей реализации ВС с учетом элементной базы и системного программного обеспечения.

В генетических алгоритмах система операций преобразования расписаний определена: операции скрещивания, мутации и селекции. Требуется выбрать такую форму представления расписаний, чтобы операции генетического алгоритма не приводили к получению недопустимых вариантов расписаний и любое допустимое расписание могло быть представлено в выбранной форме представления расписаний. В алгоритмах имитации отжига, случайного поиска, детерминированной коррекции расписаний система операций преобразования расписаний вводится при разработке алгоритма. Для этих алгоритмов способ представления расписаний и система операций преобразования расписаний должными обладать следующими свойствами: применение операции должно приводить к допустимому варианту расписания и система операций должна быть функционально полной.

Использование в итерационных алгоритмах системы операций преобразования расписаний, получающих недопустимые расписания $HP \notin HP^*_{1-5}$, приводит к необходимости введения в функцию *T=f(HP,HW)* слагаемых, отвечающих за штрафы, или барьерных функций. При этом возникают проблемы корректной оценки значений *T* для неинтерпретируемых расписаний и искажения пространства решений.

В последующих разделах предлагается решение проблемы согласованного изменения задающих расписание переменных, позволяющее избежать получения недопустимых вариантов решения на всех итерациях алгоритма, и допускающее построение итерационных алгоритмов обеспечивающих переход от произвольного допустимого расписания к оптимальному за линейное число итераций относительно числа рабочих интервалов.

4. Непосредственное представление и операции преобразования расписания

В данном разделе рассмотрим математические структуры данных для непосредственного представления расписаний (бинарное и целочисленное представления) и функционально полную систему операций для преобразования расписаний.

Бинарное непосредственное представление расписания

Расписание задается:

Матрицей привязки $Y(HP)_{N\times M}$ и матрицей смежности $X(HP)_{N\times N}$ графа *HP*, где элементы матриц определяются:

$$y_{ij} = \begin{cases} 1, \text{если } p_i \in SP_j \\ 0, \text{если } p_i \notin SP_j \end{cases} \qquad x_{ij} = \begin{cases} 1, \text{если } \prec_{ij} \in \prec_{HP} \\ 0, \text{если } \prec_{ij} \notin \prec_{HP} \end{cases}$$

(первый индекс рабочий интервал-предшественник, второй индекс рабочий интервал-приемник). Где *M* – число процессоров в ВС, *N* – число рабочих интервалов в *H*.

Недостатком данного представления является большое число бинарных переменных $N^2+N\cdot M$.

Целочисленное непосредственное представление расписания

1. Расписание задается матрицей $Y(HP)_{N\times M}$, где элемент матрицы определяется:

$$y_{ij} = \begin{cases} c, \text{если } p_i \in SP_j \\ 0, \text{если } p_i \notin SP_j \end{cases},$$

c – порядковый номер рабочего интервала p_i в SP_j.

При данном способе представления расписания число целочисленных переменных равно $N \cdot M$.

2. Расписание задается: вектором привязки $Y(HP)_K$ и вектором порядка $X(HP)_N$, где i-й элемент вектора $Y(HP)_K$ равен номеру списка в который назначены рабочие интервалы i-го процесса, а i-й элемент вектора $X(HP)_N$ равен порядковому номеру рабочего интервала в соответствующем списке. При данном способе представления расписания число целочисленных переменных равно $K+N$.

Операции преобразования расписания

Можно выделить следующие варианты отличия расписаний HP и HP' друг от друга:

- расписания HP и HP' отличаются лишь порядком рабочих интервалов как минимум в одном SP_j;
- расписания HP и HP' отличаются привязкой рабочих интервалов.

Введем соответствующие операции преобразования расписаний, позволяющие устранить указанные варианты отличия: $O=\{O1,O2\}$. Операции $O1,O2$ определим для целочисленного непосредственного представления расписания. Операции будем определять при предположениях: каждый процесс имеет не более одного рабочего интервала или при условии, что на расписание не накладывается ограничение 5. После доказательства теоремы о функциональной полноте системы операций $O=\{O1,O2\}$ и получении условий их применимости покажем возможность снятия указанных предположений.

Операция изменения порядка рабочих интервалов в одном списке изменяет порядковый номер рабочего интервала p_i в списке SP_m (порядковый номер рабочего интервала становится равным c) и корректирует порядковые номера соответствующих рабочих интервалов в данном списке (NS_m – число рабочих интервалов в списке SP_m):

$$O1(p_i, SP_m, c) \equiv \begin{cases} \left\{ c1 = y_{im},\ y_{im} = c \in (1,\ldots,c1-1); \forall j \middle|\ y_{jm} \neq 0 \vee c \le y_{jm} < c1 : y_{jm} = y_{jm} + 1 \right\} - \\ - \textit{уменьшение порядкового номера} \\ \left\{ c1 = y_{im},\ y_{im} = c \in (c1+1,\ldots,NS_m); \forall j \middle|\ y_{jm} \neq 0 \vee c1 < y_{jm} \le c : y_{jm} = y_{jm} - 1 \right\} - \\ \textit{увеличение порядкового номера} \end{cases}$$

Операция изменения привязки рабочих интервалов переносит рабочий интервал p_i из списка SP_m в список SP_k (порядковый номер рабочего интервала становится равным c) и корректирует порядковые номера соответствующих рабочих интервалов в списках SP_m и SP_k:

$$O2(p_i, SP_m \to SP_k, c) \equiv \{ y_{ik} = c \in (1 \ldots NS_k + 1); \forall j \middle|\ y_{jk} \neq 0 \vee c \le y_{jk} : y_{jk} = y_{jk} + 1,\ NS_k = NS_k + 1;$$
$$\forall j \middle|\ y_{jm} \neq 0 \vee y_{im} < y_{jm} : y_{jm} = y_{jm} - 1,\ NS_m = NS_m - 1,\ y_{im} = 0 \}$$

Указанные интервалы параметра операций c могут привести к нарушению ограничений 3,4. Ниже покажем функциональную полноту системы операций $O=\{O1,O2\}$ и получим условия их применимости (выбор значения параметра c) не нарушающие ограничений 3,4.

Теорема 1. Если HP и HP' - произвольные допустимые расписания ($HP, HP' \in HP^*_{1-4}$), то существует конечная цепочка команд $\{O_i\}_{i=1}^{K}$, $O_i \in \{O1, O2\}$, переводящая расписание HP в HP', такая, что все K промежуточных расписаний являются допустимыми и $K \le 2N$.

Доказательство. Введем понятие канонического расписания HP^0: все рабочие интервалы находятся в SP_1 и порядковые номера рабочих интервалов в SP_1 равны номерам рабочих интервалов в графе H. HP^0 является допустимым, поскольку нумерация рабочих

интервалов в графе *H* удовлетворяет условиям полной топологической сортировки. Полноту системы операций *{O1,O2}* докажем, показав, что существует стратегия (последовательность выбора рабочих интервалов, операция для каждого интервала из последовательности и значение параметра *c*) с числом шагов $K \le N$ позволяющая перевести произвольное расписание *HP* в расписание HP^0, такая, что все промежуточные расписания HP^i (полученные после выполнения отдельной операции O_i) являются допустимыми и любая операция из цепочки $\{O_i\}_{i=1}^{K}$, $O_i \in \{O1, O2\}$ является обратимой.

Применение операций *O1,O2* не может привести к нарушению ограничений 1,2 по определению операций при любой стратегии.

Расписания *HP,* HP^0 и HP^i будем представлять в ярусной форме максимальной высоты. Покажем, что следующая стратегия позволяет перевести *HP* в HP^0, не нарушая ограничений 3-4 для промежуточных расписаний HP^i:

1) выбор рабочих интервалов осуществляется в соответствии с их номерами в *H*;
2) если очередной рабочий интервал $p_i^s \in SP_1$, то применяем *O1*; если $p_i^s \notin SP_1$, то применяем *O2* (нижний индекс – номер рабочего интервала в *H*, верхний – номер яруса на котором расположен рабочий интервал);
3) для рабочего интервала p_i^s параметр *c=i*.

Рассмотрим применение данной стратегии к рабочему интервалу с номером *i=1*. Если рабочий интервал находится в SP_1 на первом ярусе, то его перенос не требуется. Если рабочий интервал находится в SP_1 на ярусе отличном от первого, то увеличиваем на *1* номера ярусов с первого яруса по ярус, предшествующий ярусу на котором находился первый рабочий интервал, и применяем к нему операцию *O1*. Полученное при этом расписание HP^1 удовлетворяет ограничению 3 в силу того, что нумерация рабочих интервалов в *H* удовлетворяет условиям полной топологической сортировки, и является ярусным, следовательно, удовлетворяет ограничению 4. Если рабочий интервал не находится в SP_1, то применяем к нему операцию *O2*, осуществляющим его перенос в SP_1 на первый ярус. Если рабочий интервал находился на ярусе отличном от первого, то аналогично, как и при применении операции *O1*, корректируем номера ярусов. Полученное при этом расписание HP^1 удовлетворяет ограничению 3 в силу того, что нумерация рабочих интервалов в *H* удовлетворяет условиям полной топологической сортировки, и является ярусным, следовательно, удовлетворяет ограничению 4. Поскольку, *HP* допустимое расписание, то операции *O1/O2* являются обратимыми (параметр *c* в обратной операции принимает старый номер рабочего интервала).

Пусть расписание HP^{i-1} является допустимым. На предыдущих шагах все предшественники *i*-го рабочего интервала перенесены на ярусы лежащие выше *i*-го яруса в SP_1. Переносим *i*-й рабочий интервал в SP_1 на *i*-й ярус, используя операцию *O1/O2*. Если *i*-й рабочий интервал находился не на *i*-м ярусе, то перед применением операции увеличиваем на 1 номера ярусов с *i*-го яруса по ярус, предшествующий ярусу на котором находился *i*-й рабочий интервал. Полученное расписание HP^i, удовлетворяет ограничениям 3 и 4, так как все предшественники *i*-го рабочего интервала находятся на ярусах выше *i*-го яруса, последователи на ярусах ниже *i*-го яруса, и граф HP^i представлен в ярусной форме максимальной высоты. Поскольку, HP^{i-1} и HP^i допустимые расписания, то операции *O1/O2* являются обратимыми (параметр *c* в обратной операции принимает старый номер рабочего интервала).

После обхода всех вершин в соответствии с используемой стратегией получим расписание HP^0. Если операции *O1/O2* применялись для всех рабочих интервалов то *K=N*. Поскольку, каждая операция из цепочки $\{O_i\}_{i=1}^{K}$, $O_i \in \{O1, O2\}$ обратима, то существует стратегия перехода от HP^0 к произвольному *HP*, такая, что все *K* промежуточных расписаний

являются допустимыми и $K \le N$. Следовательно, существует стратегия перехода от произвольного допустимого варианта расписания *HP* к произвольному допустимому варианту расписания HP', такая, что все промежуточные расписания допустимы и $K \le 2N$.

Следствие 1. Существует стратегия перехода от произвольного допустимого расписания к оптимальному расписанию, такая, что длина цепочки команд $\{O_i\}_{i=1}^{K}$, $O_i \in \{O1, O2\}$, переводящей произвольное допустимое расписание в оптимальное, не превосходит значения $2N$ ($K \le 2N$) и все K промежуточных расписаний являются допустимыми.

Условия применимости операций не нарушающие ограничений на HP

Получим интервал значений параметра *c* при применении операции *O1/O2* к рабочему интервалу p_i^s. Расписания *HP* будем представлять в ярусной форме максимальной высоты. Обозначим через $IN_i = \{p_k^l :\prec_{ki} \neq 0\}$ - множество непосредственных предшественников рабочего интервала p_i^s (всегда выполняется $k<i$ и $l<s$), $OUT_i = \{p_k^l :\prec_{ik} \neq 0\}$ - множество непосредственных последователей рабочего интервала p_i^s (всегда выполняется $k>i$ и $l>s$). Операция $lin = \max\limits_{IN_i}(l)$ - получает максимальный номер яруса, на котором расположен один из непосредственных предшественников рабочего интервала p_i^s, для рабочих интервалов, не имеющих предшественников $lin=0$ (нулевой ярус всегда пуст). Операция $lout = \min\limits_{OUT_i}(l)$ - получает минимальный номер яруса, на котором расположен один из непосредственных последователей рабочего интервала p_i^s, для рабочих интервалов, не имеющих последователей $lout=N$ (*N* – число ярусов в *HP*, для ярусной формы максимальной высоты число ярусов всегда равно числу рабочих интервалов). Тогда, рабочий интервал p_i^s может быть размещен в любой из списков SP_j на любой из ярусов из интервала $lin<s<lout$. Если выбранный ярус занят, то осуществляется коррекция ярусной формы путем соответствующего сдвига ярусов.

Пусть рабочий интервал p_i^s переносится в SP_j или изменяется его порядковый номер в этом списке. Разобьем множество SP_j на три подмножества: $PIN_j = \{p_k^l : l \ge lin,\ p_k^l \in SP_j\}$ - множество рабочих интервалов из списка SP_j, находящихся на ярусах, расположенных выше яруса *(lin-1)*; $PI_j = \{p_k^l : lin < l < lout,\ p_k^l \in SP_j\}$ - множество рабочих интервалов из списка SP_j, находящихся на ярусах $]lin, lout[$; $POUT_j = \{p_k^l : l \le lout,\ p_k^l \in SP_j\}$ - множество рабочих интервалов из списка SP_j, находящихся на ярусах, расположенных ниже яруса *(lout+1)*. Параметр *c* при размещении рабочего интервала p_i^s в SP_j может принимать следующие значения, не нарушающие ограничения на *HP* (индекс *j* для *PIN, PI, POUT* будем опускать):

	PIN	*PI*	*POUT*	*C*
1	$PIN=\varnothing$	$PI=\varnothing$	$POUT=\varnothing$	$c=1$
2	$PIN=\varnothing$	$PI=\varnothing$	$POUT\neq\varnothing$	$c=1$
3	$PIN=\varnothing$	$PI\neq\varnothing$	$POUT=\varnothing$	$1 \le c \le \max\limits_{PI}(y_{jk})+1$
4	$PIN=\varnothing$	$PI\neq\varnothing$	$POUT\neq\varnothing$	$1 \le c \le \max\limits_{PI}(y_{jk})+1$
5	$PIN\neq\varnothing$	$PI=\varnothing$	$POUT\neq\varnothing$	$c = \min\limits_{PIN}(y_{jk})+1$
6	$PIN\neq\varnothing$	$PI=\varnothing$	$POUT=\varnothing$	$c = \min\limits_{PIN}(y_{jk})+1$

7	$PIN \neq \varnothing$	$PI \neq \varnothing$	$POUT = \varnothing$	$\min_{PI}(y_{jk}) \le c \le \max_{PI}(y_{jk}) + 1$
8	$PIN \neq \varnothing$	$PI \neq \varnothing$	$POUT \neq \varnothing$	$\min_{PI}(y_{jk}) \le c \le \max_{PI}(y_{jk}) + 1$

Указанные выше интервалы значений параметра *c* не нарушающие ограничения на *HP* могут быть расширены, если ярусная форма максимальной высоты будет приведена к такому виду, что все рабочие интервалы SP_j, которые не находятся в отношении транзитивного порядка с непосредственным предшественником рабочего интервала p_i^s (находящимся на ярусе *lin*) и расположены на ярусах с меньшими номерами чем *lin*, будут перенесены на ярусы с номерами большими чем *lin*. Аналогичное преобразование ярусной формы может быть осуществлено и для непосредственного последователя рабочего интервала p_i^s, находящимся на ярусе *lout*.

Пусть непосредственным предшественником рабочего интервала p_i^s является рабочий интервал p_m^{lin}, находящийся на ярусе *lin*, и рабочий интервал p_i^s переносится в SP_j или изменяется его порядковый номер в этом списке. Пусть lin^* ярус, на котором находится транзитивный предшественник рабочего интервала p_m^{lin}, принадлежащий SP_j и с наибольшим порядковым номером в SP_j. Рабочие интервалы SP_j, находящихся между ярусами lin^* и *lin* могут быть перераспределены по ярусам, таким образом, что они будут находиться на ярусах с большими номерами, чем *lin* (при этом осуществляется соответствующее перераспределение по ярусам рабочих интервалов и других списков). Аналогичное преобразование ярусной формы осуществляется и для последователей.

Возможность снятия условий: каждый процесс имеет не более одного рабочего интервала или на расписание не накладывается ограничение 5

Поскольку, при доказательстве теоремы 1 и получении условий применимости операций *O1/O2* использована ярусная форма максимальной высоты (на каждом ярусе находится лишь один рабочий интервал), то полученные результаты легко могут быть обобщены на случаи когда: каждый процесс может иметь более одного рабочего интервала или на расписание накладывается ограничение 5. При перемещении рабочего интервала из одного списка в другой, перемещаются также и все рабочие интервалы процесса, которому принадлежит перемещаемый рабочий интервал, но при этом остаются на прежних ярусах. В результате получаем ярусную форму нового расписания, и, следовательно, полученное расписание удовлетворяет ограничениям 3-5.

5. Параметрическое представление расписания

В данном разделе рассмотрим математические структуры данных для параметрического представления расписаний, соответствующие алгоритмы восстановления расписания по его параметрическому представлению не нарушающие ограничений 1-5, введем операции изменения параметров и докажем возможность задания любого допустимого расписания параметрическим представлением с использованием приоритетов и соответствующих быстрых алгоритмов восстановления.

Параметрическое представление расписаний с использованием приоритетов

Расписание задается вектором Y_{N+K}:

$$Y_{N+K} \equiv (\bigcup_{i=1}^{K} \langle YPR \rangle_i)$$

$$\langle YPR \rangle_i \equiv (\langle YE \rangle_i \cup \langle YP \rangle)$$

$$\langle YP \rangle \equiv (\bigcup_{j=1}^{N_i} \langle YP \rangle_j)$$

K - число процессов в H, N_i – число рабочих интервалов в i-ом процессе, $N = \sum_{i=1}^{K} N_i$ - число рабочих интервалов в H, $Y \equiv (\langle y1 \rangle \bigcup \langle y2 \rangle)$ - операция построения вектора $Y = (y1, y2)$ из параметров $\langle y1 \rangle$ и $\langle y2 \rangle$.

Параметр $\langle YE \rangle_i \in [1,\ldots,K] \subset Z$ содержит номер процессора, на котором выполняются рабочие интервалы i-го процесса, т.е. параметры $\langle YE \rangle$ однозначно определяют распределение рабочих интервалов по SP (привязку). Параметры $\langle YE \rangle$ могут принимать значения от 1 до K. Если значения всех параметров $\langle YE \rangle$ равны, то все рабочие интервалы выполняются на одном процессоре, если все значения различны, то рабочие интервалы каждого процессора выполняются на своем процессоре. В силу ограничения 5, максимальное число не пустых процессоров равно числу процессов K в H.

Параметры $\langle YP \rangle \in [1,\ldots,L] \subset Z$ используются алгоритмом восстановления расписания (определение отношения полного порядка в каждом SP_i) в качестве приоритетов рабочих интервалов.

При данном способе представления расписания число целочисленных переменных равно $N+K$.

Для описания алгоритма восстановления, множество рабочих интервалов разобьем на три подмножества: $P = P1 \bigcup P2 \bigcup P3$. $P1$ - множество рабочих интервалов распределенных в SP (определен порядковый номер рабочего интервала) алгоритмом восстановления на предыдущих шагах; $P2$ - множество рабочих интервалов, у которых все предшественники принадлежат $P1$; $P3$ - множество рабочих интервалов, у которых хотя бы один из предшественников не принадлежит $P1$.

Алгоритм восстановления полного порядка рабочих интервалов в SP (A1).

1. Начальное разбиение множества P:

$P1=\varnothing$;

$P2 = \{p_j : \forall j,k \in [1,\ldots,N] \mid \prec_{jk} = \varnothing\}$ - множество рабочих интервалов в H без предшественников;

$P3 = \{p_j : p_j \in P \setminus P2\}$ - множество рабочих интервалов в H, у которых имеются предшественники.

2. Находим в $P2$ рабочий интервал с наименьшим значением параметра $\langle YP \rangle$ (если таких интервалов более одного, то выбираем интервал с наименьшим номером):

- размещаем его в конец соответствующего списка SP (номер списка определяется значением параметра $\langle YE \rangle$);
- переносим его из $P2$ в $P1$.

3. Проверяем $P3$ с целью возможности переноса рабочих интервалов в $P2$:
если есть рабочие интервалы, у которых все предшественники принадлежат $P1$, то переносим их в $P2$.

4. Если $P2 \neq \varnothing$, то к п.2, иначе завершить работу.

Утверждение 1. Алгоритм $A1$ восстановления расписания по его параметрическому представлению Y_{N+K} получает расписание $HP \in HP^*_{1-5}$, и расписание восстанавливается однозначно.

Доказательство. Ограничение 5 не нарушается в силу того, что все рабочие интервалы, принадлежащие одному и тому же процессу, имеют одно и тоже значение параметра $\langle YE \rangle$ и будут размещены (п.2 алгоритма) в один и тот же список.

Ограничения 1,2 не нарушается в силу того, что каждый рабочий интервал переносится из *P2* в соответствующий *SP*, причем перенос осуществляется лишь один раз в ходе работы алгоритма.

Выполнение ограничений 3,4 и однозначность восстановления расписания докажем показав, что алгоритм *A1* получает на каждом шаге допустимое частичное расписание и это расписание получается однозначно. Будем представлять расписание в ярусной форме максимальной высоты. Расписание, полученное после размещения первого выбранного из *P2* рабочего интервала, всегда удовлетворяет ограничениям 3,4. Множество *P2* определено однозначно (содержит рабочие интервалы без предшественников), выбор рабочего интервала из *P2* однозначен и его порядковый номер в соответствующем *SP* определен однозначно, равен 1 (п.2 алгоритма). Пусть частичное расписание, полученное после размещения *(j-1)* рабочих интервалов допустимо и однозначно. Покажем, что на *j*-м шаге рабочий интервал выбирается и размещается алгоритмом *A1* в расписание однозначно и полученное частичное расписание удовлетворяет ограничениям 3,4. Множество *P2* на *j*-м шаге алгоритма *A1* определяется однозначно рабочими интервалами, размещенными в расписание на предыдущих шагах. Выбор рабочего интервала из *P2* однозначен и его порядковый номер в соответствующем *SP* определен однозначно, максимальный порядковый номер рабочего интервала в данном *SP* плюс 1 (п.2 алгоритма). Данный рабочий интервал размещается на *j*-й ярус. Поскольку все его предшественники (в соответствии с определением множества *P2*) уже распределены на предыдущих шагах на ярусы с меньшими номерами, то отношения частичного порядка заданное в *H* не нарушается (выполняется ограничение 3). Ограничение 4 выполняется в силу того, что полученное промежуточное расписание представлено в ярусной форме. Таким образом, после выполнения алгоритмом *A1 N* шагов полученное расписание будет удовлетворять ограничениям 3,4 и порядковые номера рабочих интервалов в списках определяются однозначно.

Теорема 2. Любое допустимое расписание $HP \in HP^{*}_{1-5}$ может быть задано параметрическим представлением с использованием приоритетов (Y_{N+K}) и однозначно восстановлено алгоритмом *A1*, если допустимая верхняя граница *L* значений параметров $\langle YP \rangle$ больше или равна числу рабочих интервалов ($L \geq N$).

Д*оказательство.* Расписания могут отличаться друг от друга привязкой и порядком рабочих интервалов (смотри раздел 3). Любая допустимая привязка (распределение рабочих интервалов по *SP*) может быть задана соответствующими значениями параметров $\langle YE \rangle \in [1,\ldots,K]$ и однозначно восстановлена алгоритмом *A1*.

Возможность задания любого допустимого порядка для фиксированной привязки можно доказать методом математической индукции. Для любого произвольно выбранного SP_k можно показать, что для рабочего интервала с порядковым номером равным 1, значение параметра $\langle YP \rangle$ для этого рабочего интервала можно выбрать таким образом, что он может иметь любой допустимый порядковый номер в SP_k, который однозначно получает алгоритм *A1* (место возможного размещения рабочего интервала определяется размещением его последователей в *HP* с использованием ярусной формы *HP*). Далее полагаем, что для (*i-1)* рабочих интервалов из списка SP_k с наименьшими порядковыми номерами, можно получить любые допустимые варианты порядка, выбирая соответствующий набор значений параметров $\langle YP \rangle$. Можно показать, что для рабочего интервала с порядковым номером равным *i*, значение параметра $\langle YP \rangle$ для этого рабочего интервала можно выбрать таким

образом, что он может иметь любой допустимый порядковый номер в SP_k, который однозначно получает алгоритм *A1* (место возможного размещения рабочего интервала определяется размещением его предшественников и последователей в *HP* с использованием ярусной формы *HP*).

Выбор значений параметров $\langle YP \rangle$, позволяющих получить требуемое расписание алгоритмом восстановления *A1*, можно сделать, проведя соответствующую топологическую сортировку вершин требуемого расписания. Поскольку, в *P2* максимум может находиться *N* рабочих интервалов, то для получения любого допустимого варианта порядка может потребоваться максимум *N* различных значений параметров $\langle YP \rangle$, т.е. может потребоваться полная топологическая сортировка.

Операции преобразования расписания

При параметрическом представлении расписания с использованием приоритетов операции преобразования расписания могут быть введены как операции изменения значений параметров $\langle YE \rangle$ и $\langle YP \rangle$:

1) параметры $\langle YE \rangle$ могут принимать целочисленные значения в диапазоне *[1,...,K]*,
2) параметры $\langle YP \rangle$ могут принимать целочисленные значения в диапазоне *[1,...,L]*,
3) количество изменяемых параметров может изменяться от 1 до *N+K*.

Операции преобразования параметрической формы представления расписаний, удовлетворяющие условиям 1-3 при использовании алгоритма восстановления *A1*, будут получать расписания удовлетворяющие ограничениям 1-5, что следует непосредственно из утверждения 1. Из теоремы 2 следует, что данные операции преобразования расписания и алгоритм восстановления *A1* позволяют получить любой допустимый вариант расписания.

Параметрическое представление расписаний с использованием характеристик рабочих интервалов

Расписание задается вектором Y_K:

$$Y_K \equiv (\bigcup_{i=1}^{K} \langle YE \rangle_i)$$

Параметр $\langle YE \rangle_i \in [1,\ldots,K] \subset Z$ содержит номер процессора, на котором выполняются рабочие интервалы *i*-го процесса, т.е. параметры $\langle YE \rangle$ однозначно определяют распределение рабочих интервалов по *SP* (привязку). Параметры $\langle YE \rangle$ могут принимать значения от 1 до *K*.

При данном способе представления расписания число целочисленных переменных равно *K*.

Алгоритм восстановления *A1* вместо значений параметров $\langle YP \rangle$ в качестве приоритетов может использовать характеристики рабочих интервалов:

- вычислительную сложность рабочего интервала,
- число непосредственных последователей,
- число непосредственных предшественников,
- априорно заданные приоритеты.

Для данного способа параметрического представления расписания справедливо утверждение 1, однако, теорема 2 не выполняется. При построении алгоритмов оптимизации, использующих данный способ представления расписания, следует обращать внимание на принципиальную возможность получения расписания с требуемым качеством.

Заключение.

В работе рассмотрена проблема построения расписаний при совместном проектировании аппаратных и программных средств вычислительных систем. Основная область эффективного применения жадных алгоритмов - построение расписаний на абстрактном уровне проектирования при решении задачи оценки принципиальной возможности построения ВС при заданных ограничениях и сужение класса рассматриваемых архитектур. Основная область эффективного применения алгоритмов, основанных на методе проб и ошибок - построение расписаний на системном уровне проектирования при решении задачи определения основных параметров структуры ВС. Основная область эффективного применения алгоритмов детерминированной коррекции решений – уточнение расписаний на уровне регистровых передач при решении задачи уточнения деталей реализации ВС с учетом выбранных элементной базы и системного программного обеспечения.

Комбинаторный характер задачи построения расписания, а именно транзитивность отношения порядка, создает проблему согласованного изменения задающих расписание переменных для получения на очередной итерации алгоритма допустимого варианта расписания. Для алгоритмов, основанных на методе проб и ошибок, и алгоритмов детерминированной коррекции предложено решение проблемы согласованного изменения переменных задающих расписание, которое позволяет избежать получения недопустимых вариантов решения на всех итерациях алгоритма и допускает построение итерационных алгоритмов обеспечивающих переход от произвольного допустимого расписания к оптимальному за линейное число итераций.

Для непосредственного способа представления расписания (привязка и порядок задаются явно) получена функционально полная система операций преобразования расписаний. Данные операции используются в алгоритмах случайного поиска, имитации отжига и детерминированной коррекции решений.

Для параметрического представления расписания (привязка и порядок задаются значениями некоторых параметров, по которым, с использованием алгоритма восстановления, их значения могут быть восстановлены) доказана возможность представления любого допустимого варианта расписания и однозначность его восстановления. Данная форма представления используется в генетических алгоритмах. Параметрическое представление допускает независимое изменение значений всех переменных в заданных интервалах, что позволяет гарантированно получать допустимые варианты расписания при выполнении операций генетического алгоритма. Данная форма представления может быть использована также и в алгоритмах случайного поиска и имитации отжига.

Литература.

1. Смелянский Р.Л. Модель функционирования распределенных вычислительных систем, // Вестн. Моск. Ун-та. сер 15, Вычисл. Матем. и Кибернетика. 1990, No. 3, стр. 3-21.
2. Смелянский Р.Л. Об инварианте поведения программ // Вестн. МГУ, сер. 15, Вычислительная математика и Кибернетика, 1990., No. 4, С. 54-60.
3. В.В. Воеводин. Математические модели и методы в параллельных процессах. – М.: Наука, 1986.
4. Теория расписаний и вычислительные машины/ Под ред. Э.Г.Коффмана. М.: Наука, 1984. - 334с.
5. Гэри М., Джонсон Д. Вычислительные машины и трудно решаемые задачи. - М.: Мир, 1982. - 416с.
6. Беллман Р. Динамическое программирование. – М.: ИЛ, 1960.
7. М. Мину. Математическое программирование. Теория и алгоритмы.- М.: Наука, 1990.
8. Т. Кормен, Ч. Лейзерсон, Р. Ривест. Алгоритмы: построение и анализ. – М.: МЦНМО, 1999.
9. Костенко В.А. Метод автоматизированного синтеза параллельных программ для вычислительных систем с MIMD архитектурой// Программирование, 1993., No. 1, стр. 43-57.
10. Костенко В.А. Автоматизация составления параллельных программ для вычислительных систем с MIMD архитектурой// Кибернетика и системный анализ, 1995., No. 5, стр. 170-179.
11. Костенко В.А., Романов В.Г., Смелянский Р.Л. Алгоритмы минимизации аппаратных ресурсов ВС// Искусственный интеллект (Донецк), 2000., No2, С.383-388.

12. Костенко В.А. Задачи синтеза архитектур: формализация, особенности и возможности различных методов для их решения// Программные системы и инструменты. Тематический сборник. - М.: МАКС Пресс, 2000., № 1, С.31-41.
13. Батищев Д.И., Гудман Э.Д., Норенков И.П., Прилуцкий М.Х. Метод комбинирования эвристик для решения комбинаторных задач упорядочивания и распределения ресурсов// Информационные технологии, 1997., N 2 - С.29-32.
14. Норенков И. П. Эвристики и их комбинации в генетических методах дискретной оптимизации// Информационные технологии, 1999, N 1.
15. Holland J.N. Adaptation in Natural and Artificial Systems. Ann Arbor, Michigan: Univ. Michigan Press, 1975.
16. Zbigniev Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs// Third, Revised and Extended Edition – Springer, 1999.
17. Костенко В.А. Принципы построения генетических алгоритмов и их использование для решения задач оптимизации// Труды IV Международной конференции "Дискретные модели в теории управляющих систем" (19-25 июня 2000 г.) – М.: МАКС Пресс, 2000., С.49-55.
18. Обозрение прикладной математики. Серия “Методы оптимизации”. Эволюционные вычисления и генетические алгоритмы// Т.3, выпуск 5. - 1996.
19. Уоссермен Ф. Нейрокомпьютерная техника. Теория и практика.- М.: Мир, 1992.
20. Костенко В.А. Алгоритмы оптимизации, опирающиеся на метод проб и ошибок, в совместном проектировании аппаратных и программных средств ВС// Труды Всероссийской научной конференции "Высокопроизводительные вычисления и их приложения" (30 октября - 2 ноября 2000 г., г. Черноголовка). - М.: Изд-во МГУ, 2000., С.123-127.
21. Л.А. Растригин. Статистические методы поиска.- М.: Наука, 1968.

Приложение 1.
Набор состояний рабочих интервалов процессов и аппаратных ресурсов.

Состояния рабочего интервала процесса:

- *SP_INI* - ожидание инициализации;
- *SP_D* – ожидание завершения предшественников;
- *SP_EAS* - ожидание получения разделяемого ресурса;
- *SP_W* - работа (выполнение внутренних действий *SP_WE* или внешних взаимодействий *SP_WEAS*);
- *SP_SPR* – ожидание обновления данных в памяти процессора или завершения, вызванного системного процесса;
- *SP_END* - рабочий интервал выполнен.

Состояния процессора:

- *SE_W* - выполняет один из рабочих интервалов, назначенных на этот процессор;
- *SE_EX* - простой из-за нахождения текущего рабочего интервала в состояниях *SP_D* или *SP_EAS*;
- *SE_SPR* – выполняется системный процесс;
- *SE_END* - завершение выполнения всех рабочих интервалов назначенных на процессор.

Состояния разделяемого ресурса:

- *SEAS_INI* - арбитраж заявок на обслуживание и инициализация выполнения заявки, выигравшей арбитраж;
- *SEAS_W* - обслуживание заявки процессора;
- *SEAS_EX* - простой из-за отсутствия заявок на обслуживание от процессоров.