Оптимальный алгоритм локального планирования инструкций при отсутствии структурных конфликтов

С. С. Сурин*
sergey.surin@pobox.spbu.ru

В статье рассматривается задача построения оптимального по критерию времени расписания вычисления набора выражений на процессоре, работающем в конвейерном режиме. Набор выражений не должен содержать общих подвыражений, а все операции имеют одну и ту же длительность выполнения процессором, равную t тактам, где t — любое положительное число. Выясняется структура оптимальных расписаний и предлагается алгоритм решения этой задачи, имеющий полиномиальную сложность.

Введение

Задачи планирования работы современных ЭВМ в большинстве являются NP-трудными (см. [3, 4, 5, 6]). Поэтому для решения многих из них строятся эвристические алгоритмы. Тем больший интерес представляют полиномиальные алгоритмы, дающие оптимальные решения даже для узких классов таких задач.

Вычислительная система, для которой мы будем решать задачу планирования ее работы, состоит из одного процессора и памяти двух видов: основной и регистровой. Объем памяти обоих видов считается неограниченным.

* НИИ математики и механики им. ак. В. И. Смирнова СПбГУ, 198504, Университетский пр., 28. Санкт-Петербург, Россия

Процессор работает в конвейерном режиме и выполняет арифметические и логические операции с одним и двумя аргументами и операции обмена данными между основной и регистровой памятью. Обмен данными между операциями происходит только через регистровую память. Результат каждой арифметической и логической операции помещается в регистр. Каждая арифметическая и логическая операция имеет несколько модификаций. В одной модификации все ее аргументы перед началом выполнения операции должны быть загружены в регистры. В других модификациях все или некоторые аргументы могут быть литералами, которые указываются в записи операции.

Все операции имеют одинаковую длительность выполнения, измеряемую тактами. Эту величину мы обозначим через t.

Аппаратных ограничений на последовательность выполнения операций не имеется.

Мы будем изучать организацию работы процессора при вычислении значений набора выражений. Набор выражений не содержит общих подвыражений и задается графом зависимостей по данным, который является лесом.

Требуется составить расписание выполнения вычислительной системой операций набора, по которому он вычислялся бы за минимальное время.

Использованный в статье математический аппарат (теория графов и теория расписаний) можно найти в [1, 2].

1. Набор выражений: основные понятия

Пусть C — произвольный набор выражений. Через $G(C)$ обозначим его граф зависимостей по данным. Если операция не использует результатов других операций, то назовем ее *листовой*, а в противном случае — *нелистовой* операцией. Если результат операции не используется другими операциями набора (это значит, что результат данной операции является значением одного из выражений набора), то соответствующую ей вершину графа $G(C)$ будем называть *вершиной выхода*.

Назовем длиной пути, соединяющего две вершины графа $G(C)$, число входящих в него дуг. Удаленностью данной вершины p от множества вершин выхода назовем длину самого длинного пути с началом в вершине p и концом в вершине выхода[1]. Удаленность

[1]Если граф $G(C)$ является лесом, то из каждой его вершины, которая не

вершины выхода от множества вершин выхода по этому определению равна нулю.

Нам будет удобно упорядочить все операции набора выражений C в виде некоторого списка, который мы будем называть списком A. Структура этого списка определяется графом $G(C)$ следующим образом. В списке A операции разделены на группы. В одну группу входят все операции, которым соответствуют вершины, равноудаленные от множества вершин выхода. Число групп обозначим через k_{max}. В списке группы располагаются в порядке уменьшения удаленности. Таким образом, первую группу списка составляют листовые операции, соответствующие вершинам графа $G(C)$, наиболее удаленным от множества вершин выхода, а последнюю — все операции, соответствующие вершинам выхода. Упорядоченность входящих в одну группу операций — произвольная (эти операции не связаны зависимостями по данным).

На рис. 1 приведен пример списка A для набора из двух выражений. В списке A символом $L(X)$ обозначена операция загрузки в регистр значения переменной X, находящегося в основной памяти. Список состоит из пяти групп ($k_{max} = 5$).

Когда процессор выполняет операции в конвейерном режиме, интервал времени вычисления набора выражений можно разбить на две части. Первая часть содержит первые такты выполнения (то есть стартовые такты) всех операций набора. Эту часть мы будем называть стартовой частью интервала вычисления.

В течение второй части интервала завершают свое выполнение последние операции набора. Так как все операции имеют одну и ту же длительность выполнения, равную t тактам, то длительность второй части всегда равна $t - 1$ тактам.

Задать расписание вычисления набора выражений значит указать для каждой его операции стартовый такт.

Такты стартовой части интервала вычисления, в которых по данному расписанию не стартует ни одна операция, мы будем называть *пустыми* тактами.

Если операция p имеет единственный вычисляемый аргумент, то обозначим через $c_1(p)$ операцию, которая завершает его вычисление. Если операция p имеет два вычисляемых аргумента, то операции, завершающие их вычисление, обозначим через $c_1(p)$ и $c_2(p)$.

является вершиной выхода, выходит только один путь с концом в вершине выхода.

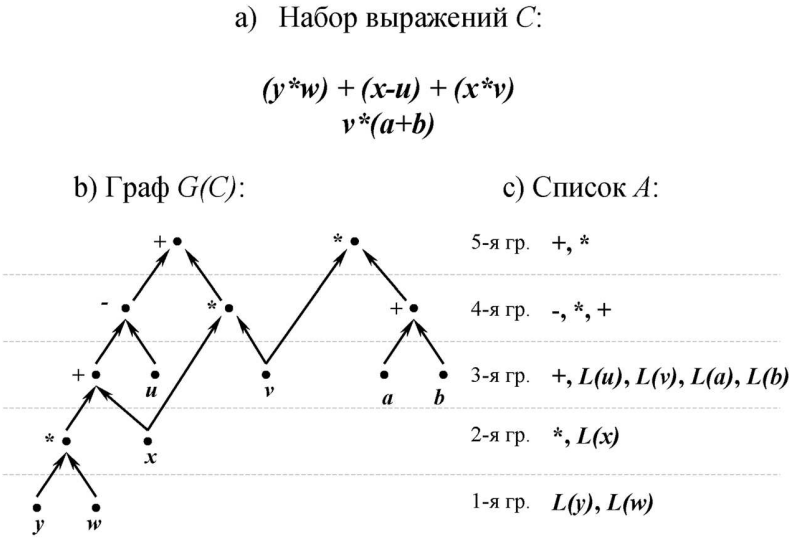

Рис. 1. Набор выражений, его граф и список A

Операцию, которая завершает вычисление последнего по времени аргумента операции p, мы будем называть операцией, завершающей подготовку операции p к выполнению, и обозначать через $c(p)$.

Листовая операция p может начать свое выполнение в любом такте стартовой части интервала вычисления. Нелистовая может стартовать только после того, как завершится ее операция $c(p)$.

Такты, следующие за последним тактом выполнения операции $c(p)$, будем называть *тактами готовности* операции p к выполнению и обозначать через $\theta(p)$. Первый из этих тактов будем обозначать через $\theta_1(p)$.

Мы будем говорить, что расписание допустимо по времени, если по нему каждая нелистовая операция p стартует в одном из тактов $\theta(p)$.

Так как расписания, которые не являются допустимыми по времени, не выполнимы, то далее термином «расписание» мы будем обозначать только допустимые по времени расписания.

На рис. 2 приведено расписание вычисления набора выражений C с рис. 1. Длительность выполнения всех операций равна трем тактам. Через `NOP` обозначается пустой такт. Легко проверить до-

```
          L y,r1;       L w,r2;
L x,r3;   NOP;          M r1,r2,r1;
L u,r2;   L v,r4;       A r1,r3,r1;   L a,r5;       L b,r6;
NOP;      NOP;          S r1,r2,r1;   M r3,r4,r3;   A r5,r6,r5;
NOP;      A r1,r3,r1;   M r4;r5;r4;
```

Рис. 2. Расписание s_1, $t = 3$

пустимость этого расписания.

Оптимизация расписания по времени означает поиск такого расписания, по которому вычисление набора выражений выполнялось бы за наименьшее время.

Длительность интервала вычисления линейно (с коэффициентом 1) зависит от числа пустых тактов в его стартовой части. Поэтому оптимальным по времени расписанием мы будем называть такое расписание, по которому стартовая часть интервала вычисления содержит минимальное число пустых тактов.

Так как в каждом такте может стартовать не более одной операции, то в каждом расписании можно указать k_{max} тактов, в которых стартуют последние операции всех групп списка A. Этими тактами стартовая часть интервала вычисления разбивается на подынтервалы, которые мы будем называть *стартовыми интервалами* групп списка A (или короче — стартовыми интервалами). Набор операций, стартующих в этих тактах, мы будем называть *критическим набором* операций данного расписания.

Критический набор в расписании s_1 состоит из следующих пяти операций:

```
L w,r2
M r1,r2,r1
L b,r6
A r5,r6,r5
M r4,r5,r4
```

Стартовыми тактами этих операций стартовая часть интервала вычисления разбивается на пять стартовых интервалов. В записи расписания s_1 операции, стартующие в одном стартовом интервале, стоят в одной строке.

Пусть S — расписание вычисления произвольного набора выражений C. Пронумеруем группы списка A в порядке их расположения в этом списке, а порожденные расписанием S стартовые интер-

валы в порядке следования во времени. Тогда стартовый интервал k-й группы получит номер k. Из определения стартового интервала следует, что при любом k ни одна операция k-й группы не может стартовать в стартовом интервале с номером больше k. Упорядочим операции критического набора по порядку следования их стартовых тактов.

Теорема 1. *Каково бы ни было расписание вычисления набора выражений, длительность каждого стартового интервала, кроме первого, не может быть меньше t тактов, то есть длительности выполнения операций. Длительность 1-го стартового интервала не может быть меньше числа операций в 1-й группе списка A.*

Доказательство. Пусть p — операция критического набора, входящая в k-ю группу списка A, причем $k < k_{max}$. Следовательно, она стартует в последнем такте k-го стартового интервала и при этом завершает подготовку к выполнению хотя бы одной операции $(k+1)$-й группы списка A, например операции q.

Операция q может стартовать в $(k+1)$-м стартовом интервале не раньше его t-го такта. Если она стартует в t-м такте этого интервала, причем стартует последней из операций своей группы, то длительность $(k+1)$-го стартового интервала равна t тактам, а в противном случае больше этой величины.

Первая группа списка A содержит только листовые операции. По каждому расписанию все они должны стартовать в 1-м стартовом интервале. Возможны расписания, по которым в этом интервале стартуют операции только первой группы. Листовые операции могут стартовать подряд без пустых тактов. Следовательно, длительность 1-го стартового интервала не меньше числа операций в 1-й группе списка A. □

Фрагментом расписания назовем такую его часть, в которой указаны стартовые такты всех операций, стартующих в одном или нескольких следующих подряд друг за другом стартовых интервалах. Эти интервалы будем называть стартовыми интервалами, принадлежащими данному фрагменту расписания.

Будем говорить, что расписание распадается на два фрагмента, если выполнены условия:

- первому фрагменту принадлежат первые $k_{f_1} < k_{max}$ стартовых интервалов, а второму — остальные $k_{max} - k_{f_1}$ интервалов;

```
                L y,r1;        L w,r2;
L x,r3;         L v,r4;        M r1,r2,r1;
L u,r2;         L a,r5;        A r1,r3,r1;    L b,r6;
M r3,r4,r3;     S r1,r2,r1;    A r5,r6,r5;
NOP;            A r1,r3,r1;    M r4;r5;r4;
```

Рис. 3. Расписание s_2

- ни одна операция из группы с номером больше k_{f_1} списка A не стартует в стартовом интервале с номером не больше k_{f_1}.

Теорема 2. *(Достаточные условия оптимальности расписания по времени.) Любое расписание оптимально по времени, если оно распадается на два фрагмента, удовлетворяющие условиям: стартовые интервалы, принадлежащие первому фрагменту, не содержат пустых тактов, а длительность каждого стартового интервала, принадлежащего второму фрагменту, равна t тактам.*

Доказательство. Суммарная длительность стартовых интервалов, принадлежащих по отдельности каждому из фрагментов — минимальна.

После объединения фрагментов в одно расписание улучшить его невозможно. Действительно, единственным средством его улучшения мог бы стать перенос старта операций из стартовых интервалов одного фрагмента в стартовые интервалы другого.

Но перенос из интервалов первого фрагмента в интервалы второго невозможен, так как, по определению разбиения расписания на фрагменты, ни одна операция из групп с номерами больше k_{f_1} не стартует в стартовых интервалах с номерами не больше k_{f_1}.

Перенос же в обратном направлении только ухудшит расписание, так как после него длительность интервала-отправителя не уменьшится, а длительность интервала-получателя возрастет. □

Расписание s_1 на рис. 2 в нашем примере не оптимально. В нем можно удалить два пустых такта. В результате получается расписаниеs s_2, приведенное на рис. 3.

Это расписание улучшить уже нельзя. Оно распадается на два фрагмента. Первому фрагменту принадлежат первые четыре стартовых интервала ($k_{f_1} = 4$). Второму фрагменту принадлежит последний, 5-й стартовый интервал.

2. Набор выражений, не содержащих общих подвыражений

В Теореме 2 сформулированы достаточные условия оптимальности по критерию времени расписания вычисления любого набора выражений при одинаковой длительности выполнения всех операций. Если же выражения не содержат общих подвыражений, то эти условия становятся не только достаточными, но и необходимыми условиями оптимальности. Мы не будем останавливаться на доказательстве этого факта, а приведем алгоритм построения расписания и докажем, что если выражения не содержат общих подвыражений, то это расписание удовлетворяет условиям Теоремы 2, откуда последует его оптимальность.

Если набор C выражений не содержит общих подвыражений, то, за исключением операций последней группы списка A, чьи результаты используются за пределами интервала вычисления, результат каждой операции используется в качестве аргумента только одной операцией. Множество наборов, не содержащих общих подвыражений, обозначим через C_T.

Итак, пусть C — набор выражений из множества C_T.

Поскольку в нашей вычислительной системе имеется только один процессор, то в каждом такте может стартовать только одна операция. Все операции имеют одну и ту же длительность выполнения. Поэтому в каждом такте может завершать свое выполнение не более чем одна операция.

Поскольку набор C не содержит общих подвыражений, то каждая его операция может завершать подготовку к выполнению не более одной операции. Если такт τ является последним тактом выполнения операции p, которая завершает подготовку к выполнению операции q, то такт $\tau + 1$ становится тактом θ_1 операции q. Из этих рассуждений следует, что по любому расписанию вычисления набора выражений $C \in C_T$ первые такты готовности к выполнению нелистовых операций различны. Напомним, что каждая листовая операция готова стартовать в любом такте интервала вычисления.

Пусть S — расписание вычисления набора выражений $C \in C_T$. Если в этом расписании любая пара операций p и $c(p)$ стартуют в разных стартовых интервалах, то мы будем называть его *каноническим расписанием* и множество таких расписаний обозначать через S_c.

Пусть S — каноническое расписание вычисления набора $C \in C_T$.

Обозначим через $m(S,k)$ число операций, стартующих по расписанию S в k-м стартовом интервале. Стартовый интервал с номером k мы будем называть

- *переполненным*, если $m(S,k) > t$ или $k = 1$;
- *полным* или *неполным* в зависимости от того, имеет ли место $m(S,k) = t$ или $m(S,k) < t$ соответственно.

Теорема 3. *Пусть k и $(k+1)$ — номера двух стартовых интервалов, причем в расписании $S \in S_c$ в k-м интервале стартует меньше операций, чем в $(k+1)$-м. Тогда в $(k+1)$-м интервале стартует, по меньшей мере одна операция, которая готова стартовать и в k-м.*

Доказательство. Так как каждая операция набора выражений C имеет один или два аргумента, а ее результат может использоваться в качестве аргумента только одной операцией $C \in C_T$, то в $(k+1)$-м стартовом интервале стартуют не более $m(S,k)$ операций, использующих результаты операций, стартующих в k-м интервале.

Поскольку $S \in S_c$ и $m(S,k) < m(S,k+1)$, то в $(k+1)$-м интервале стартуют, по меньшей мере, $m(S,k+1) - m(S,k)$ операций, не зависящих от результатов других операций, стартующих в k-м и $(k+1)$-м интервалах.

Каждая из этих операций может быть либо листовой, либо нелистовой операцией p, чья операция $c(p)$ стартует до начала k-го интервала. Так как все операции имеют длительность выполнения t тактов, а длительность стартового интервала не может быть меньше t тактов (Теорема 1), то в обоих случаях эти операции могут стартовать в k-м стартовом интервале. □

3. Алгоритм построения расписания вычисления набора выражений, не содержащих общих подвыражений

Следующий очень простой алгоритм (полиномиальной сложности) строит оптимальное по времени расписание вычисления набора выражений, не содержащих общих подвыражений.

В процессе работы алгоритма список A корректируется. Поэтому текущим списком в описании алгоритма мы будем называть результат его последней коррекции и обозначать через A_c. Начальным состоянием списка A_c является список A.

```
S(A)
{
  A_c = A;
  for v = 1 while A_c ≠ ∅ do
    if в A_c есть операция p, готовая стартовать в такте v
    then begin
      назначить в такте v старт операции p;
      удалить v из A_c
    end
    else считать v пустым;
    v = v + 1
  done
}
```

Рис. 4. Алгоритм S

Будем считать, что все такты интервала вычисления пронумерованы числами $1, 2, \ldots$ Через v обозначим номер текущего такта.

Обозначим расписание, построенное алгоритмом S, через S_a и покажем, что оно оптимально по времени. Для этого убедимся, что оно удовлетворяет условиям Теоремы 2.

Через $N(k)$ обозначим множество операций k-й группы списка A, которые входят в список A_c перед началом поиска операции, готовой к выполнению в первом такте k-го стартового интервала. Через $n(k)$ обозначим число этих операций. Перед началом формирования k-го стартового интервала операции множества $N(k)$ занимают первые места в списке A_c. В течение формирования этого интервала алгоритм удаляет их из списка A_c. Процесс формирования заканчивается с удалением последней из них.

Лемма 1. *Если $n(k) \leq t$, то в расписании S_a длительность k-го стартового интервала равна t тактам.*

Доказательство. Пока алгоритм S строит k-й стартовый интервал, операции множества $N(k)$ проверяются на готовность к выполнению в первую очередь. Все они становятся готовыми к выполнению в течение первых t тактов k-го интервала. Первые такты готовности их к выполнению различны, причем последняя из них становится готовой к выполнению в t-м такте k-го стартового интервала. Поэтому алгоритм назначает старт всех этих операций именно в t

первых тактах k-го интервала, причем t-й такт становится его последним тактом. □

Лемма 2. *Если $n(k) \geq t$, то в расписании S_a k-й стартовый интервал не содержит пустых тактов и состоит из стартовых тактов операций только k-й группы списка A.*

Доказательство. Из определения стартового интервала следует, что все операции k-й группы списка A, еще не стартовавшие до начала k-го стартового интервала, то есть операции множества $N(k)$, должны стартовать в этом интервале.

Выделим во множестве $N(k)$ подмножество N_1 операций, у каждой из которых такт θ_1 совпадает с одним из первых t тактов k-го стартового интервала. Остальные операции (обозначим их множество через N_2) становятся готовыми к выполнению до начала k-го интервала. Таким образом, в каждом из t первых тактов k-го интервала готовы стартовать все операции множества N_2, а в некоторых из них еще и по меньшей мере одна из операций множества N_1. Поскольку $|N1| + |N2| = n(k) \geq t$, то в множестве $N(k)$ найдутся t операций, которые смогут стартовать подряд в t первых тактах k-го интервала. Все остальные $n(k) - t$ операций готовы стартовать в каждом из последних $n(k) - t$ тактов k-го интервала.

Так как в течение формирования k-го стартового интервала еще не удаленные из списка A_c операции множества $N(k)$ остаются в нем на первых местах, то для каждого такта этого интервала готовую стартовать в нем операцию алгоритм S найдет именно среди них. □

Замечание. Все операции 1-й группы списка A являются листовыми операциями и перед началом формирования 1-го стартового интервала в списке A_c занимают первые места. Поэтому в расписании S_a этот интервал не содержит пустых тактов и состоит из стартовых тактов операций только 1-й группы. Напомним, что по определению он является переполненным интервалом независимо от числа стартующих в нем операций.

Лемма 3. *Каждый переполненный в расписании S_a стартовый интервал с номером k не содержит пустых тактов и состоит из стартовых тактов операций только k-й группы списка A.*

Доказательство. При $k = 1$ утверждение леммы следует из замечания.

Пусть $k > 1$. Так как $m(S_a, k) > t$, то длительность k-го интервала больше t тактов.

Из Леммы 1 следует, что $n(k) > t$, и Лемма 3 следует из Леммы 2. □

Из Лемм 1 и 3 следует, что расписание S_a — каноническое и что полные и неполные стартовые интервалы имеют длительность t тактов.

Лемма 4. *Все операции, стартующие в неполном по расписанию S_a стартовом интервале после первого пустого такта, стартуют в первых тактах готовности к выполнению.*

Доказательство. Пусть k — номер неполного стартового интервала в расписании S_a. Пусть все его такты пронумерованы числами $1, 2, \ldots, t$, и τ_0 — номер первого пустого такта в нем.

Обозначим через $A_c(\tau)$ состояние списка A_c перед началом поиска алгоритмом S операции, готовой стартовать в такте τ k-го интервала.

Пусть τ_1 — номер такта из диапазона $[\tau_0 : t]$, в котором стартует операция (обозначим ее через p_1), и такого, что такт $(\tau_1 - 1)$ — пустой.

Тот факт, что такт с номером $(\tau_1 - 1)$ пустой, означает, что в нем не готова стартовать ни одна из операций списка $A_c(\tau_1 - 1)$ и что этот список содержит только нелистовые операции. Следовательно, $\tau_1 = \theta_1(p_1)$, причем p_1 — единственная из операций списка $A_c(\tau_1)$, которая готова стартовать в этом такте, так как такты θ_1 нелистовых операций различны.

Пусть теперь в тактах с номерами $\tau_1, \tau_1 + 1, \ldots, \tau_1 + m - 1$, $m > 1$ стартуют операции $p_1, p_2, \ldots, p_m$, причем $\tau_1 + m - 1 = t$, или такт $\tau_1 + m$ — пустой.

Для операции p_2 такт $\tau_1 + 1$ является ее тактом θ_1, так как она не была готова стартовать в предыдущих тактах. При этом она — единственная в списке $A_c(\tau_1 + 1)$ операция, готовая стартовать в этом такте.

Индукцией по номеру такта можно доказать, что и остальные операции $p_3, \ldots, p_m$ стартуют в первых тактах готовности к выполнению.

Если $\tau_1 + m - 1 = t$, то доказательство закончено.

Если $\tau_1 + m - 1 < t$, то аналогичными рассуждениями можно убедиться в справедливости утверждения леммы относительно каждой следующей операции, стартующей в k-м стартовом интервале. □

Обозначим через k_{01} номер последнего по расписанию S_a переполненного стартового интервала. При $k_{01} = 1$ справедливость следующих лемм вытекает из замечания. Поэтому положим $k_{01} > 1$.

Лемма 5. *Первые k_{01} стартовых интервалов по расписанию S_a не содержат пустых тактов.*

Доказательство. Предположим, напротив, что среди первых k_{01} стартовых интервалов есть такие, которые содержат пустые такты. Обозначим через k номер последнего из них, $k \leq k_{01}$. Но так как k_{01}-й интервал — переполненный и потому не содержит пустых тактов (Лемма 3), то k-й интервал — неполный, и, следовательно, $k < k_{01}$.

Пусть k-й интервал содержит $n > 0$ пустых тактов и стартовые такты $t - n < t$ операций.

По выбору k-го стартового интервала следующий, $(k + 1)$-й, не содержит пустых тактов, и потому в нем стартуют не меньше t операций, то есть больше операций, чем в k-м.

По Теореме 3 в $(k + 1)$-м интервале есть операции, старт которых можно перенести в k-й интервал. Обозначим через p первую из таких операций в списке A_c.

Покажем, что алгоритм S должен предоставить операции p для старта один из тактов k-го интервала. Это будет означать, что число пустых тактов в этом интервале должно быть равно не n, а $n - 1$.

Обозначим через τ_0 первый пустой такт, а через τ_1 последний такт k-го интервала.

Если операция p листовая, или если она нелистовая, и в расписании S_a $\theta_1(p) \leq \tau_0$, то алгоритм S назначит ее старт в такте τ_0.

Если $\tau_0 < \theta_1(p) < \tau_1$, то из Леммы 4 следует, что такт $\theta_1(p)$ совпадет с одним из следующих пустых тактов k-го интервала, в котором и будет назначен старт операции p в этом случае.

Таким образом, предположив, что $n > 0$, мы приходим к противоречию. Следовательно, единственное возможное значение n есть 0, но это и есть утверждение леммы. □

Лемма 6. *По расписанию S_a ни одна операция из группы списка A с номером больше k_{01} не стартует ни в одном стартовом интервале с номером не больше k_{01}.*

Доказательство. Пусть, напротив, операции из групп с номерами больше k_{01} стартуют в стартовом интервале с номером k_2 не больше k_{01}. Обозначим эти операции символами $p_1, p_2, \ldots$. Пусть они стартуют в тактах с номерами $\tau_1 < \tau_2 < \ldots$.

Из Лемм 3 и 5 следует, что k_2-й интервал может быть только полным стартовым интервалом, а отсюда следует, что $k_2 < k_{01}$.

Интервал с номером k_2 можно выбрать так, чтобы он был последним по времени интервалом с номером меньше k_{01}, в котором стартуют операции из групп списка A с номерами больше k_{01}. Тогда в интервалах с номерами $k_2 + 1, \ldots, k_{01}$ будут стартовать операции из групп с номерами только из этого же диапазона.

Через n обозначим число операций $p_1, p_2, \ldots$, стартующих в выбранном таким образом k_2-м интервале, и покажем, что вместо по меньшей мере одной из операций $p_1, \ldots, p_n$ в k_2-м интервале по расписанию S_a стартуют операции из групп списка A с номерами не больше k_{01}. Из этого противоречия и будет следовать утверждение леммы.

Допустим, что алгоритм S в своей работе сделает исключение и не станет искать операции, готовые стартовать в тактах $\tau_1, \ldots, \tau_n$, оставив их пустыми. Тогда он построит расписание S', которое совпадает с расписанием S_a в стартовых интервалах с номерами не больше k_{01}, за исключением k_2-го интервала.

Совпадение расписаний в интервалах с номерами меньше k_2 очевидно, так как действия алгоритма при их построении для обоих расписаний совпадают.

Покажем, что по этой же причине расписания S_a и S' совпадают и в интервалах с номерами $k_2 + 1, \ldots, k_{01}$.

Стартовые интервалы и группы списка A с номерами из диапазона $[k_2 + 1 : k_{01}]$ будем для краткости называть $k_2 + 1$-интервалами и $k_2 + 1$-группами.

При построении расписания S_a перед началом формирования $k_2 + 1$-интервалов первые места в списке A_c занимают не получившие к этому моменту стартовых тактов операции всех $k_2 + 1$-групп. В расписании S_a в $k_2 + 1$-интервалах стартуют операции только $k_2 + 1$-групп, и эти интервалы не содержат пустых тактов. Это означает, что алгоритм S, формируя при построении расписания S_a $k_2 + 1$-интервалы, просматривал в списке A_c операции только $k_2 + 1$-групп и находил среди них готовые стартовать операции.

При построении расписания S' перед началом формирования $k_2 + 1$-интервалов первые места в списке A_c занимают те же опе-

рации k_2 + 1-групп. Отсюда следует, что действия алгоритма по формированию k_2 + 1-интервалов при построении обоих расписаний совпадают.

В k_2-м интервале по расписанию S' стартуют $(t-n)$ операций, то есть меньше, чем в k_2+1-м, который является полным или переполненным интервалом. По Теореме 3 в (k_2+1)-м интервале стартуют операции, готовые стартовать и в k_2-м интервале. Первую из них в списке A_c обозначим через q.

Операция q в списке A_c расположена ближе к началу, чем операции $p_1, \ldots, p_n$. Поэтому если алгоритм S не будет делать упомянутого выше исключения и будет искать операцию, готовую стартовать в очередном из тактов с номерами $\tau_1, \ldots, \tau_n$, то операцию q он найдет раньше, чем эти операции.

Далее точно так же, как в доказательстве Леммы 5 показывается, что алгоритм S предоставит операции p такт для старта в k-м интервале, здесь можно показать, что операции q он предоставит такт для старта в k_2-м интервале. □

Теорема 4. *Расписание S_a, построенное алгоритмом S, оптимально по времени.*

Доказательство. Ясно, что если по расписанию S_a стартовая часть интервала вычисления не содержит пустых тактов, то оно оптимально по времени.

Пусть теперь в стартовой части есть пустые такты.

Из Леммы 6 следует, что расписание S_a распадается на два фрагмента: первому фрагменту принадлежат первые k_{01} стартовых интервалов, а второму — остальные $k_{max} - k_{01}$.

По Лемме 5 интервалы, принадлежащие первому фрагменту, не содержат пустых тактов.

Так как k_{01} — номер последнего переполненного интервала, то длительность интервалов, принадлежащих второму фрагменту, равна t тактам.

Таким образом, условия Теоремы 2 выполнены, и, следовательно, расписание S_a — оптимально по времени. □

Заключение

Рассмотренная нами частная задача построения расписания работы Вычислительной Системы является полиномиально трудной

задачей. Тот факт, что нам удалось не только найти очень простой и эффективный алгоритм ее решения, но и охарактеризовать структуру этого решения, является скорее исключением, чем правилом при изучении этой задачи в общей постановке.

Тем не менее алгоритм S можно использовать и в более сложных задачах планирования работы вычислительных систем, получая эффективные эвристические, а в некоторых случаях и точные алгоритмы их решения.

Так, например, этот алгоритм, дополненный процедурой переупорядочивания списка A, которая выполняется на некоторых его шагах и не нарушает его полиномиальности, дает оптимальное расписание в задаче с длительностями выполнения операций 1 и 2 такта.

Список литературы

[1] *Романовский И. В.* Алгоритмы решения экстремальных задач. — М.: Наука, 1977. — 352 с.

[2] *Танаев В. С., Шкурба В. В.* Введение в теорию расписаний. — М.: Наука, 1975. — 256 с.

[3] *Garey M. R., Johnson D. S.* Computers and Intractability: A Guide to the Theory of NP-Completeness. — New York: W. H. Freeman and Company, 1979.

[4] *Hennessy J. L., Gross T. R.* Code generation and reorganization in the presence of pipeline constraints // Conference Record of the 9th Annual ACM Symposium on Principles of Programming Language. — New York: ACM. — 1982. — P. 120–127.

[5] *Lawler E., Lenstra J. K., Martel C.* e. a. Pipeline scheduling: A survey // Computer Science Research Rep. — Calif.: IBM Research Division. San Jose. — 1987.

[6] *Palem K., Simons B.* Scheduling rime-critical instructions on RISC machines // Conference Record of the 17th Annual ACM Symposium on Principles of Programming Language. — New York: ACM. — 1990.— P. 270–280.