Оптимальное локальное планирование инструкций с длительностью выполнения 1 и 2 такта при отсутствии 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. Процессор выполняет одно- и двухтактные операции, а набор выражений не содержит общих подвыражений. Выясняется структура оптимальных расписаний и предлагается алгоритм решения задачи, имеющий полиномиальную сложность.

Введение

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

Процессор работает в конвейерном режиме и может выполнять арифметические и логические операции с одним и двумя аргументами и операции обмена данными между основной и регистровой памятью. Объем памяти обоих видов считается неограниченным.

Каждая арифметическая и логическая операция имеет несколько модификаций. В одной модификации все ее аргументы перед выполнением должны находиться в регистрах. В других модификациях все или некоторые аргументы могут быть литералами, которые указываются в записи операции. Результат каждой такой операции помещается в регистр.

Длительность выполнения операций измеряется тактами. По этому признаку все операции делятся на две группы: однотактные и двухтактные операции. Для краткости первые мы будем называть *1-операциями*, а вторые — *2-операциями*.

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.

Требуется составить расписание выполнения вычислительной системой операций набора, по которому он вычислялся бы за минимальное время[1].

Частный случай этой задачи был изучен в [4].

В [2] рассматривается задача, которая отличается от нашей лишь тем, что процессор выполняет операции одной длительности t тактов. Предложенный там алгоритм решения задачи имеет полиномиальную сложность. Дополнив этот алгоритм процедурой предварительного упорядочивания входной информации, которая также имеет полиномиальную сложность, мы превратили его в алгоритм решения нашей задачи. Поэтому некоторые части данной статьи почти дословно повторяют соответствующие фрагменты текста [2]. Это сделано для удобства ее чтения и указывает на единство подхода к решению обеих задач.

Использованный нами математический аппарат можно найти в [1, 3].

1. Структура оптимального расписания

Множество наборов выражений, не содержащих общих подвыражений, обозначим через C_T.

Пусть $C \in C_T$ —набор выражений.

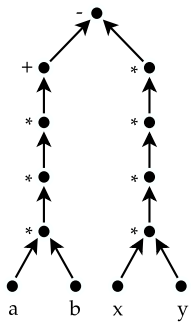
Через $G(C)$ обозначим граф зависимостей по данным этого набора. Граф $G(C)$ является лесом.

Если операция набора C не использует результатов других опе-

[1] Данная задача была предложена Д. Ю. Булычевым.

(a) Набор выражений C
*a*b*2*3+1-x*y*5*6*7*

(b) Граф



(c) Список A:
1-я гр.: L(a), L(b), L(x), L(y)
*2-я гр.: *, **
*3-я гр.: *, **
*4-я гр.: *, **
*5-я гр.: +, **
6-я гр.: -

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

раций, 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*.

Операции, соответствующие вершинам, образующим путь в графе $G(C)$, будем называть операциями, *связанными* с этим путем.

Удаленностью данной вершины p от множества вершин выхода назовем число дуг, входящих в путь с началом в вершине p и концом в вершине выхода. Так как граф $G(C)$ — лес, то из p выходит единственный путь $s(p)$, ограниченный вершиной выхода. Если p — вершина выхода, то $s(p) = (p)$, и поэтому удаленность p от множества вершин выхода равна нулю.

Нам будет удобно представить 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.

Операции, входящие в одну группу исходного списка A, располагаются в произвольном порядке. Предлагаемый нами алгоритм решения задачи построения расписания начинает свою работу с упорядочивания операций в группах с помощью специальной процедуры.

На рис. 1 приведен пример списка A для набора, состоящего из одного выражения. В списке A символом $L(X)$ обозначена операция загрузки в регистр значения переменной X, находящегося в основной памяти. Операции сложения $(+)$ и вычитания $(-)$ выполняются за один такт (являются 1-операциями). Операции загрузки в регистр (L) и умножения $(*)$ выполняются за два такта (являются 2-операциями). Список состоит из шести групп ($k_{max} = 6$).

Когда процессор выполняет операции в конвейерном режиме, интервал времени вычисления набора выражений можно разбить на две части. Первая часть содержит первые такты выполнения, т. е. *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 в нашей задаче одна операция выполняется за один или за два такта, то второй части интервала вычисления либо может не быть, либо ее длительность составит один такт.

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

Такты интервала вычисления, в которых по данному расписанию не стартует ни одна операция, мы будем называть *пустыми* тактами. В частности, пустым мы будем считать последний такт интервала вычисления, если этот такт образует его вторую часть.

Пусть S — расписание вычисления набора выражений C.

Если операция p имеет вычисляемые аргументы, то операцию, которая по расписанию S стартует последней из операций, принимающих участие в вычислении этих аргументов, 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

```
L a,r1;     L b,r2;     L x,r3;     L y,r4;
M r1,r2,r1; M r3,r4,r3;
M r1,2,r1;  NOP;        M r3,5,r3;
M r1,3,r1;  M r3,6,r3;
A r1,1,r1;  M r3,7,r3;
NOP;        S r1,r3,r1;
```

Рис. 2. Расписание s_1

стартовать только после того, как завершит свое выполнение 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 назовем тактом $\theta_1(p)$.

Для листовой операции все такты стартовой части интервала вычисления являются тактами готовности к выполнению, причем тактом θ_1 для нее можно считать первый такт интервала вычисления.

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 допустимое по времени расписание.

Пример допустимого расписания приведен на рис. 2. По расписанию s_1 может быть вычислено выражение C, представленное на рис. 1. Через `NOP` обозначается пустой такт. Легко проверить допустимость этого расписания.

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

Длительность интервала вычисления зависит только от числа имеющихся в нем пустых тактов. Поэтому *оптимальным по времени* расписанием мы будем называть такое расписание, по которому интервал вычисления содержит минимальное число пустых тактов.

Расписание s_1 не является оптимальным: в нем слишком много пустых тактов. Расписание s_2 на рис. 3 действительно оптимально. В нем стартовая часть интервала вычисления не содер-


```
L a,r1;     L b,r2;     L x,r3;     L y,r4;
M r1,r2,r1; M r3,r4,r3;
M r1,2,r1;  M r3,5,r3;
M r1,3,r1;  M r3,6,r3;
M r3,7,r3;  A r1,1,r1;
S r1,r3,r1;
```

Рис. 3. Расписание s_2

жит ни одного пустого такта, а длительность второй части равна нулю.

Итак, для вычисления набора выражений $C \in C_T$ требуется построить расписание с минимальным числом пустых тактов в интервале вычисления.

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

Стартовыми тактами операций, стартующих по расписанию S последними по времени среди операций своих групп списка A, разобьем интервал вычисления на части, которые назовем *стартовыми интервалами*. Пронумеруем группы в порядке их расположения в списке A, а стартовые интервалы в порядке их следования во времени. Тогда каждая группа и стартовый интервал, в последнем такте которого стартует ее операция, получат один и тот же номер. Стартовый интервал с номером k назовем стартовым интервалом k-й группы списка A. При любом $k : 1 \le k < k_{max}$ ни одна операция k-й группы не может стартовать в стартовом интервале с номером больше k.

Обозначим через $t(p)$ время выполнения операции p. Функция $t(p)$ принимает значения 1 и 2. Через $T(S,k)$ обозначим длительность стартового интервала с номером k по расписанию S.

Пусть $1 \le k \le k_{max}$ и p_k — операция, стартующая по расписанию S в последнем такте k-го стартового интервала. Положим $T_{min}(S,1) = 0$, и $T_{min}(S,k) = t(p_{k-1})$ при $k > 1$.

При каждом $k = 1, \ldots, k_{max}$ $T(S,k) \ge T_{min}(S,k)$, так как по любому расписанию S в 1-м стартовом интервале стартуют операции 1-й группы списка A, а при $k > 1$ операция q, для которой $p_{k-1} = c(q)$, может стартовать не раньше $t(p_{k-1})$-го такта k-го стартового интервала.

Расписание, по которому каждая нелистовая операция p и ее операция $c(p)$ стартуют в разных стартовых интервалах, будем на-

зывать *каноническим* расписанием и множество таких расписаний обозначать через S_c.

Через $m(S,k)$ обозначим число всех операций, стартующих по расписанию S в k-м стартовом интервале.

Пусть $S \in S_c$. Стартовый интервал с номером k будем называть: *полным*, если $T(S,k) = T_{min}(S,k) = m(S,k)$, *неполным*, если $T(S,k) = T_{min}(S,k) > m(S,k)$, и *переполненным* интервалом, если $T(S,k) = m(S,k) > T_{min}(S,k)$.

Из этого определения следует, что в каждом неполном интервале стартует только одна операция, а в каждом переполненном — две или больше операций. Пустой такт, причем только один, может содержать только неполный интервал.

Расписание $S \in S_c$ будем называть *нормализованным* расписанием и множество их обозначать через S_n, если каждый стартовый интервал по этому расписанию является либо полным, либо неполным, либо переполненным интервалом.

Любое каноническое, но не нормализованное расписание можно преобразовать в нормализованное, удалив из интервала вычисления лишние пустые такты и изменив при необходимости порядок старта операций внутри некоторых стартовых интервалов.

Теорема 1. *Пусть $S \in S_n$, $C \in C_T$, и k_1 и $k_2 > k_1$ — номера двух стартовых интервалов. Пусть по расписанию S k_1-й интервал — неполный, а в k_2-ом интервале стартуют две или больше операций. При этом если $k_2 > k_1 + 1$, то пусть в интервалах с номерами $k_1 + 1, \ldots, k_2 - 1$ стартуют по одной операции. Тогда в k_2-м интервале стартует по меньшей мере одна операция, готовая стартовать и в пустом такте k_1-го интервала.*

Доказательство. Так как результат каждой операции набора выражений C может использоваться в качестве аргумента только одной операцией, то в k_2-м стартовом интервале стартуют $m(S,k_2) - 1$ операций, не использующих результат операции, стартующей в $k_2 - 1$-м интервале, и, следовательно, готовых стартовать в первом такте $k_2 - 1$-го интервала (длительность каждой операции не больше двух тактов!). Поэтому существует расписание $S' \in S_n$, по которому одна из этих операций, например, операция p из группы списка A с номером не меньше k_2, стартует в первом такте $(k_2 - 1)$-го интервала (во втором, последнем, такте этого интервала должна стартовать операция из $(k_2 - 1)$-й группы).

Если $k_2 = k_1 + 1$, то теорема доказана.

Если $k_2 > k_1 + 1$, то повторим эти рассуждения для пар интервалов с номерами $(k_2 - 1, k_2 - 2), \ldots, (k_1 + 1, k_1)$ и получим расписание $S'' \in S_n$, по которому операция p будет стартовать в первом пустом такте k_1-го стартового интервала. □

Пусть $S \in S_n$ и $C \in C_T$.

Пусть числа k' и k'' таковы, что $0k' < k''k_{max}$. При этом, если $k' > 0$, то пусть k'-й интервал — переполненный, а если $k'' < k_{max}$, то и k''-й интервал — также переполненный. Если $k' + 1 < k''$, то пусть интервалы с номерами $k' + 1, \ldots, k'' - 1$ — не переполненные интервалы.

Если по расписанию S в интервалах с номерами из диапазона $[k' + 1 : k'']$ стартуют все операции из всех групп списка A с номерами только из этого же диапазона, то часть расписания S, определяющую стартовые такты операций в этих интервалах, мы будем называть *фрагментом* расписания S. Обозначим эту часть расписания через F и положим $k_1(F) = k' + 1$ и $k_2(F) = k''$.

Стартовые интервалы и группы списка A с номерами $k_1(F), \ldots, k_2(F)$ будем называть интервалами и группами фрагмента F.

Если интервал с номером $k_2(F)$ по расписанию S — переполненный, то фрагмент F назовем *невырожденным*, а в противном случае (что возможно только при $k_2(F) = k_{max}$) — *вырожденным* фрагментом. Следовательно, вырожденным может быть только последний фрагмент расписания.

Отрезок пути в графе $G(C)$ назовем *стержневым* путем фрагмента F, если связанные с ним операции стартуют во всех стартовых интервалах этого фрагмента. Каждый фрагмент может иметь не более двух стержневых путей.

Стержневой путь $s = (p_1, \ldots, p_r)$ фрагмента F назовем *опорным* путем этого фрагмента, если операция p_1 является нелистовой операцией.

Полный интервал с номером k фрагмента F, для которого $T_{min}(S,k) = 2$, будем называть *1-2-интервалом*, если по расписанию S в нем стартуют 1-операция и 2-операция, связанные с опорными путями этого фрагмента.

Фрагмент F расписания $S \in S_n$ назовем *идеальным* фрагментом, если:

α) из того, что $k_2(F)$-й интервал — переполненный, следует, что интервалы фрагмента F не содержат пустых тактов;

β) из того, что $k_2(F)$-й интервал — переполненный, и в группе с номером $k_2(F)$ списка A есть 1-операции, следует, что одна из них стартует в последнем такте этого интервала;

γ) из того, что среди интервалов фрагмента есть 1-2-интервал, следует, что в последнем такте этого интервала стартует 1-операция.

Из этого определения следует, что вырожденный фрагмент является идеальным фрагментом, если удовлетворяет только условию γ. Интервалы такого фрагмента могут содержать пустые такты.

Расписание $S \in S_n$ назовем *идеальным* расписанием, если оно состоит из идеальных фрагментов.

Расписание s_2 на рис. 3 является идеальным расписанием. Оно состоит из двух идеальных фрагментов. Первому фрагменту принадлежит только 1-й стартовый интервал, который по всем расписаниям является переполненным интервалом. Второй фрагмент — вырожденный. Ему принадлежат остальные пять стартовых интервалов (со 2-го по 6-й). Он имеет два опорных пути. С одним путем связаны операции, стартующие в первых тактах интервалов с номерами $2, \ldots, 5$. С другим связаны операции, стартующие во вторых тактах этих интервалов. Операция, стартующая в 6-м интервале связана с обоими путями. 5-й интервал является 1-2-интервалом.

Теорема 2. *Любое идеальное расписание вычисления набора выражений $C \in C_T$ оптимально по времени.*

Доказательство. Пусть S — идеальное расписание. Обозначим через k_{01} номер последнего переполненного по этому расписанию интервала.

Из определения фрагмента следует, что в стартовых интервалах с номерами не больше k_{01} стартуют все операции каждой группы списка A с номером не больше k_{01}, и не стартует ни одна операция из групп с номерами больше k_{01}.

Из условия α идеальности фрагмента следует, что ни один интервал с номером не больше k_{01} не содержит пустых тактов, а условия β и γ обеспечивают минимальное суммарное время *выполнения* стартующих в этих интервалах операций.

Если $k_{01} = k_{max}$, то теорема доказана.

Если $k_{01} < k_{max}$, то последний идеальный фрагмент расписания S является вырожденным фрагментом. Это значит, что каждый из

стартовых интервалов с номером больше k_{01} является полным, либо неполным интервалом, а условия β и γ гарантируют, что длительность каждого из них будет минимальной из возможных. Если стартовый интервал с номером k_{max} является 1-2-интервалом, то условие γ обеспечивает отсутствие второй части интервала вычисления.

Таким образом, по отдельности обе части расписания S оптимальны. Улучшить это расписание невозможно, так как перенос стартовых тактов операций из интервалов первой части в интервалы второй противоречит определению стартового интервала, а перенос в обратном направлении лишь ухудшит расписание. □

Согласно определению фрагмента в интервалах каждого идеального фрагмента идеального расписания стартуют все операции всех групп этого фрагмента.

Из определения опорного пути фрагмента следует, что по идеальному расписанию операции, связанные с таким путем, стартуют в стартовых интервалах своих групп списка A.

Из определения 1-2-интервала следует, что идеальный фрагмент может иметь только один такой интервал.

Пусть F_1 и F_2 — два соседних идеальных фрагмента идеального расписания S: $k_1(F_2) = k_2(F_1) + 1$. Пусть интервал с номером k, $k_1(F_2)kk_2(F_2)$, является 1-2-интервалом.

Обозначим через q_1 1-операцию, а через q_2 2-операцию, стартующие в k-м интервале. Операция q_1 стартует в его втором такте. Через s_1 и s_2 обозначим опорные пути фрагмента F_2, с ними связаны операции q_1 и q_2: $s_1 = (p_1^1, \ldots, q_1, \ldots)$, $s_2 = (p_1^2, \ldots, q_2, \ldots)$. Если $k = k_1(F_2)$, то $s_1 = (q_1)$ и $s_2 = (q_2)$.

Из определения 1-2-интервала следует, что интервалы с номерами $k_1(F_2), \ldots, k$ — полные двухтактные. Для операции, стартующей во втором такте такого интервала, этот такт является тактом θ_1 (в противном случае интервал был бы переполненным). Отсюда следует, что если операция q_1 стартует во втором такте k-го интервала, то все операции, связанные с путем s_1 и стартующие до операции q_1, являются 2-операциями и стартуют во вторых тактах своих интервалов. Верно и обратное: если все эти операции, кроме q_1, являются 2-операциями и стартуют во вторых тактах своих интервалов, то операция q_1 стартует во втором такте k-го интервала. То же самое можно сказать об операции q_2 и операциях, связанных с путем s_2.

Теорема 3. *Для того, чтобы по идеальному расписанию S второй такт 1-2-интервала с номером k был для 1-операции q_1 тактом θ_1, необходимо и достаточно, чтобы операция $c(p_1^1)$ была 2-операцией и стартовала в последнем такте $k_2(F_1)$-го интервала.*

Доказательство. Так как $(k_2(F_1)+1)$-й интервал — полный двухтактный, то в последнем такте $k_2(F_1)$-го интервала может стартовать либо операция $c(p_1^1)$, либо операция $c(p_1^2)$, причем обе эти операции должны быть 2-операциями.

Если в этом такте стартует операция $c(p_1^2)$, то во вторых тактах интервалов с номерами $k_1(F_2),\dots,k$ стартуют операции, связанные с путем s_2, и, следовательно, операция q_1 не может стартовать во втором такте k-го интервала.

Если в последнем такте $k_2(F_1)$-го интервала стартует операция $c(p_1^1)$, то во вторых тактах интервалов с номерами $k_1(F_2),\dots,k$ стартуют операции, связанные с путем s_1, и, в частности, операция q_1 стартует во втором такте k-го интервала, причем этот такт является для нее тактом θ_1, так как k-й интервал — полный. □

Группу списка A назовем *смешанной* группой, если она содержит 1-операции и 2-операции, *1-группой*, если содержит только 1-операции, и *2-группой*, — если только 2-операции.

Теорема 4. *Группа списка A с номером $k_2(F_1)$ является 2-группой.*

Доказательство. Фрагмент F_1 является невырожденным идеальным фрагментом. Поэтому $k_2(F_1)$-й интервал является переполненным интервалом. По теореме 3 в последнем такте этого интервала стартует 2-операция. Из условия β следует, что это возможно лишь тогда, когда $k_2(F_1)$-я группа списка A не содержит 1-операций. □

2. Алгоритм построения расписания вычисления набора выражений

Следующий алгоритм (полиномиальной сложности) строит оптимальное по времени расписание вычисления набора выражений, не содержащих общих подвыражений. Он состоит из двух блоков: процедуры Ord, упорядочивающей список A, и процедуры S, которая по упорядоченному списку A строит расписание.


```
Ord(A) {
    for k= 1 while k <= k_max do
      if A[k] — смешанная группа then
          в A[k] на первые места поместить все 2-,
          а за ними все 1-операции
      else
       if A[k] — 2-группа then begin
          в графе G(C) выделить множество S путей
          с началами из A[k] и концами в 1-вершинах;
          if S не пусто then begin
             в S выбрать самый короткий путь s_1;
             начало s_1 поместить в A[k] на последнее место
          end
      end
      k= k+1
   done
}

S(A) {
    выполнить процедуру Ord с аргументом список A;
    A_c= A;
    for v= 1 while A_c ≠ 0 do
       if в A_c есть операция p, готовая стартовать в такте v
       then begin
          назначить в такте v старт операции p;
          удалить эту операцию из списка A_c
       end
       else считать такт v пустым;
       v=v+1
   done
}
```

Рис. 4. Алгоритм построения расписания

В процедуре Ord через $A[k]$ обозначена k-я группа списка A.

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

Будем предполагать, что все такты интервала вычисления пронумерованы числами $1, 2, \dots$

На рис. 4 представлен алгоритм построения расписания.

Обозначим расписание, построенное процедурой S, через S_a и покажем, что оно оптимально по времени.

Будем предполагать, что процедура S просматривает список A_c, начиная всегда с первого элемента и в качестве операции p выбирает первую в этом списке подходящую операцию.

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 интерва-

ла. Через $n(k)$ будем обозначать число этих операций. Перед началом формирования k-го стартового интервала операции множества $N(k)$ занимают первые места в списке A_c. Поэтому процедура S в поисках операции, готовой стартовать в очередном такте k-го интервала, в первую очередь проверяет операции множества $N(k)$. Найденные операции она удаляет из списка A_c. Процесс формирования k-го стартового интервала заканчивается с удалением последней операции множества $N(k)$.

Лемма 1. *Расписание S_a — нормализованное.*

Доказательство. Пусть $n(k)T_{min}(S_a, k)$. Так как процедура S текущий такт v предоставляет первой найденной в списке A_c операции и после этого удаляет ее из списка, то в данном случае $T_{min}(S_a, k)$-й такт k-го стартового интервала она предоставит последней оставшейся в списке A_c операции множества $N(k)$, и этот интервал станет полным или неполным интервалом.

Если $n(k) > T_{min}(S_a, k)$, то для каждого такта k-го интервала процедура S найдет в множестве $N(k)$ операцию, готовую стартовать в этом такте, и интервал окажется переполненным. □

Замечание 1. *В переполненном по расписанию S_a интервале с номером k стартуют операции только k-й группы списка A.*

Лемма 2. *Переполненные по расписанию S_a интервалы разбивают это расписание на фрагменты.*

Доказательство. Если расписание S_a таково, что при всяком k, $1 \le k < k_{max}$, при котором k-й стартовый интервал является переполненным интервалом, ни одна операция из группы с номером больше k не стартует в интервале с номером не больше k, то утверждение леммы справедливо.

Предполагая, что лемма не верна, обозначим через k_0 номер переполненного интервала, для которого указанное свойство расписания S_a нарушено, а через k_1k_0 — номер интервала, в такте τ которого стартует операция p из группы списка A с номером $k_1 > k_0$.

Интервал с номером k_1 — полный двухтактный, так как, согласно замечанию 1, по расписанию S_a в переполненном интервале операции нескольких групп не стартуют. Поэтому $k_1 < k_0$. Числа k_1 и k_0 выберем так, чтобы интервалы с номерами $k_1 + 1, \ldots, k_0 - 1$ (если они есть) не были переполненными, и в них стартовали бы только операции из групп списка A с номерами не больше k_0.

Через k_2 обозначим номер первого после k_1-го интервала, в котором стартуют две или больше операций: k_2k_0.

Если предположить, что такт τ является пустым, а не стартовым тактом операции p, то к интервалам с номерами k_1 и k_2 можно применить теорему 1, согласно которой в k_2-м интервале стартует операция q, готовая стартовать в такте τ. Согласно выбору чисел k_0, k_1 и k_2, она входит в группу с номером не больше k_0, и, следовательно, в списке A_c находится ближе к его началу, чем операция p. Поэтому такт τ процедура S предоставит именно ей. □

Лемма 3. *Каждый фрагмент расписания S_a удовлетворяет условию α идеальности фрагмента.*

Доказательство. Эта лемма доказывается по той же схеме, как и лемма 2. Предполагая, что в расписании S_a есть фрагмент F, для которого условие α нарушено, мы увидим, что у этого фрагмента найдутся два интервала: неполный с номером k_1 и интервал с номером $k_2 > k_1$, содержащий стартовые такты двух или большего числа операций, причем к этим интервалам применима теорема 1. Из этого следует, что процедура S найдет в списке A_c операцию (по крайней мере, из $k_2(F)$-й группы списка A), готовую стартовать в пустом такте k_1-го интервала. □

Замечание 2. *Можно показать, что если процедура S предоставила такт для старта в k_1-м интервале операции из группы списка A с номером $k_2 > k_1$, стоящей в списке A_c последней из операций этой группы, то по расписанию S_a в k_2-м интервале стартует только одна операция, и, следовательно, он не может быть переполненным интервалом.*

Лемма 4. *Каждый фрагмент расписания S_a удовлетворяет условию β идеальности фрагмента.*

Доказательство. Пусть F — произвольный невырожденный фрагмент расписания S_a.

Если $k_2(F)$-я группа списка A — 1-группа, то лемма очевидна.

Если эта группа — смешанная, то процедура Ord все ее 1-операции ставит в ней на последние места. Перед началом построения $k_2(F)$-го интервала $n(k_2(F)) > 1$, поскольку этот интервал — переполненный, и с учетом замечания 2 можно утверждать, что последняя операция множества $N(k_2(F))$ (обозначим ее через p) является 1-операцией.

Легко проверить, что первые $T_{min}(S_a, k_2(F))$ тактов $k_2(F)$-го интервала процедура S предоставит первым $T_{min}(S_a, k_2(F))$ операциям множества $N(k_2(F))$. Следующие такты для старта в этом интервале операции получат в порядке своего расположения в множестве $N(k_2(F))$. В частности, последний такт интервала получит операция p. □

Лемма 5. *Каждый фрагмент расписания S_a удовлетворяет условию γ идеальности фрагмента.*

Доказательство. Пусть F_1 и F_2 — два соседних фрагмента расписания S_a. Пусть интервал с номером k является 1-2-интервалом фрагмента F_2. Пусть $s = (p_1, \ldots)$ — опорный путь фрагмента F_2, с которым связана 1-операция q, стартующая в k-м интервале.

По теореме 4 $k_2(F_1)$-я группа списка A — 2-группа, и, следовательно, она упорядочивалась процедурой Ord. На $k_2(F_1)$-м шаге работы этой процедуры единственным самым коротким путем, ограниченным 1-вершиной (а именно, вершиной q), был путь с началом в вершине графа $G(C)$, соответствующей операции $c(p_1)$, входящей в $k_2(F_1)$-ю группу списка A. Поэтому именно эту операцию процедура Ord поставила на последнее место в группе.

Так как $k_2(F_1)$-й интервал по расписанию S_a — переполненный, то из замечания 2 следует, что процедура S предоставит операции $c(p_1)$ для старта последний такт этого интервала. По теореме 3 этого достаточно для того, чтобы второй такт k-го интервала был тактом θ_1 для операции q.

Перед началом построения k-го интервала операция q будет первой в списке A_c операцией, готовой стартовать в его втором такте, и, следовательно, этот такт процедура S предоставит для старта именно ей. □

Теорема 5. *Расписание S_a оптимально по времени.*

Доказательство. Из лемм 1-5 следует, что S_a — идеальное расписание. Согласно теореме 2, всякое идеальное расписание оптимально по времени. Следовательно, таковым является и расписание S_a. □

Заключение

Рассмотренная нами задача (далее (1,2)-задача) получается из задачи, изученной в [2] (далее t-задачи), модификацией вычислительной системы, а именно изменением длительностей выполнения операций. Можно обобщить это изменение, положив, что какие-то одни операции выполняются за t тактов (t-операции), а остальные — за $t+1$ тактов ($(t+1)$-операции).

Можно модифицировать систему в другом направлении: например, изменять в ней число процессоров. При этом в (1,2)-задаче (или вообще в $(t, t+1)$-задаче) можно рассмотреть две модификации. В одной все процессоры идентичны. В другой часть процессоров выполняют только t-операции, а остальные — только $(t+1)$-операции.

Увеличение числа процессоров в t-задаче оставляет ее полиномиально сложной. Простая корректировка алгоритма S из [2], дающая ему возможность назначать старт операций не на одном, а на нескольких процессорах, делает его алгоритмом решения обобщенной t-задачи.

Появление столь же простых алгоритмов в других обобщениях, по нашему мнению, вряд ли возможно.

Список литературы

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

[2] *Сурин С. С.* Оптимальный алгоритм локального планирования инструкций при отсутствии конфликтов // Системное программирование. Вып. 1. Изд-во СПбГУ, 2005. С. 131–146.

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

[4] *Kurlander S. M., Proebsting T. A., Fischer C. N.* Efficient Instruction Scheduling for Delayed-Load Architectures // ACM TOPLAS. Vol. 17. N 5. 1995. P. 740–776.