Шкурко Д. В.

О СЛОЖНОСТИ ПОСТРОЕНИЯ ОПТИМАЛЬНОГО ЛИНЕЙНОГО УЧАСТКА*

1. ВВЕДЕНИЕ

Оптимизация линейных участков занимает важное место в построении оптимизирующих трансляторов и, как правило, включается в набор оптимизирующих преобразований, выполняемых транслятором. От оптимизации линейных участков требуется, чтобы ее сложность была приемлемой, так как это всего лишь один из видов оптимизаций, производимых транслятором. К сожалению, этого можно достичь не всегда, и приходится выделять некоторый набор оптимизирующих приемов, который, возможно, не дает лучшего участка относительно данных преобразований и критерия оптимальности.

Как правило, при оптимизации программ интересуются двумя критериями: временем исполнения и объемом требуемой памяти. В данной работе внимание сконцентрировано на первом критерии.

Рассматриваемая модель линейных участков, введенная в [4], является расширением моделей из [1] и [6]. В модели из [1] рассматриваются только схемные преобразования, в [6] добавляется единственная интерпретированная операция пересылки. В модели [4] добавляются конструкции для описания структурных переменных, и в качестве интерпретации используются классы эквивалентности термов, чтобы описать повторные присваивания частям структурных переменных. Благодаря тому, что эта модель более выразительна, она потенциально лучше подходит для использования в реальных трансляторах. Среди существенных загрублений модели следует отметить отсутствие учета косвенной адресации ([2]).

Из соображений эффективности применения в модели из [4] были определены преобразования, имеющие явный оптимизирующий эффект. Аналогичный подход используется в [3], где, помимо полного набора преобразований, выделяются частные случаи общих преобразований, позволяющие эффективно искать возможности по их применению.

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

До конца не было ясно, какова сложность построения оптимального участка в рассматриваемой модели. В [5] была описана процедура оптимизации для сокращенного набора преобразований. В данной статье показана NP-полнота задачи оптимизации в общем случае. Такая сложность сохраняется и для более простой модели из [6].

2. МОДЕЛЬ ЛИНЕЙНЫХ УЧАСТКОВ И.В. ПОТТОСИНА

Фиксируется счетное множество Σ имен переменных и конечное множество Θ операций с заданнной арностью. В подмножестве Θ выделяется подмножество $\Theta' \subset \Theta$ легковычислимых функций. Пересылка θ_0 лежит в Θ'. Отдельно выделяется конечное множество функций выбора Λ.

Линейным участком, или *блоком,* называется тройка $\mathcal{B} = (P, V, U)$, где

- P — конечный список операторов $S_0; S_1; \ldots; S_n; S_{n+1}$ $(n \geq 0)$,
- $V \subset \Sigma$ — конечное множество *входных* переменных,
- $U \subset \Sigma$ — конечное множество *выходных* переменных.

Переменные, входящие в операторы и не лежащие в подмножествах U и V, называются *промежуточными.* Множество таких переменных обозначается W. Элементы множеств Σ, U, V, $U \cup V$ и W обозначаются далее, соответственно, буквами x, u, v, z и w с подходящими индексами.

Оператор S_0 — $start(v_1, \ldots, v_k)$, где $v_1, \ldots, v_k$ — все входные переменные. Оператор S_{n+1} — $finish(u_1, \ldots, u_k)$, где $u_1, \ldots, u_k$ — все выходные переменные. Остальные операторы имеют вид

$$A \leftarrow \theta B_1 \ldots B_r,$$

где $\theta \in \Theta$ и $B_1, \ldots, B_r$ — переменные или выборы аргумента (см. ниже), а A — переменная или выбор результата (см. ниже). Говорят, что B_i сопоставлены входам, а A — выходу.

Структурные переменные вводятся с помощью двух конструкций. Конструкция вида

$$g(z_1, \ldots, z_l)[z_{l+1} \ldots z_{l+k}], g \in \Lambda$$

называется *выбором аргумента,* если она сопоставлена входу; или *выбором результата,* если она сопоставлена выходу. g — функция выбора, $z_{l+1} \ldots z_{l+k}$ — *список аргументов* или *список результатов,* соответственно. Если переменная входит в список аргументов или результатов,

то считается, что в оператор она входит неявно, иначе — вхождение явное.

Если выходу оператора S_i сопоставлена переменная, то эта переменная является *обязательным* результатом, иначе, если сопоставлен выбор результата, множеством *необязательных* результатов является список результатов.

Аргументы оператора — это все переменные, сопоставленные входам, в объединении со всеми переменными из выборов аргументов и всеми переменными из выбора результата, неявляющимися результатами.

Отдельно определяются аргументы и результаты операторов S_0 и S_{n+1}. Обязательные результаты оператора *start* — все входные переменные, аргументов у *start* нет. Аргументы оператора $finish$ — все выходные переменные, результатов у $finish$ нет.

Аргументы оператора S_i обозначаются A_i, обязательные результаты — R'_i, а все результаты — $R_i \supset R'_i$.

Линейный участок задает вычисление некоторого терма для каждой выходной переменной. Линейные участки эквивалентны, если для каждой выходной переменной они вычисляют один и тот же терм. Этот терм задается с помощью интерпретации всех конструкций в множестве слов над алфавитом $V \cup \Theta \cup \Lambda$.

- В результате исполненения оператора *start* все входные переменные получают значения: $\mathrm{Val}_0(A) = A, A \in V$. Значения остальных переменных неопределеные.
- Функции выбора, входящей в S_i, сопоставляется слово

$$\mathrm{Val}_i\, g(z_1, \ldots, z_n) = g\, \mathrm{Val}_{i-1}\, z_1 \ldots \mathrm{Val}_{i-1}\, z_n.$$

- Входу оператора S_i сопоставляется $\mathrm{Val}_{i-1}(A)$, если вход — это переменная A, либо

$$\mathrm{Val}_i\, g(z_1, \ldots, z_l)[\mathrm{Val}_{i-1}\, z_{l+1} \ldots \mathrm{Val}_{i-1}\, z_{l+k}],$$

  если вход является выбором аргумента.
- Наконец, определяется, каким образом изменяются значения переменных после исполнения оператора $S_i = A \leftarrow \theta B_1 \ldots B_r$. Обозначим $\mathrm{Val}\, S_i = \theta\, \mathrm{Val}_{i-1}(B_1) \ldots \mathrm{Val}_{i-1}(B_r)$.
  - Если A — переменная, то $\mathrm{Val}_i(A) = \mathrm{Val}\, S_i$.
  - Если $A = g(z_1, \ldots, z_l)[z_{l+1} \ldots z_{k+l}]$ — выбор результата, то пусть $t_1 = \mathrm{Val}_i\, g(z_1, \ldots, z_l)$, $t_2 = \mathrm{Val}\, S_i$. Тогда, при условии, что подстрока вида jt_1t_2 не содержится в $\mathrm{Val}_{i-1}(z_{k+j})$,

$\mathrm{Val}_i\, z_{k+j} = \mathrm{Val}_{i-1}\, z_{k+j} \# j t_1 t_2$, в противном случае значение z_{k+j} не меняется.

Значения других переменных не меняются. Пересылка не меняет значений: $\theta_0 B = B$.

- Если значения каких-то аргументов оператора неопределены, то и значение оператора неопределено.
- Другим способом переменная получить значение не может.

Требуется, чтобы значения всех выходных переменных были определены на выходе.

Переменная A *активна* после некоторого оператора S_t, если

- $A \in R_i$,
- $A \notin R_{i+1}, \ldots, R_j$,
- $0 \le i \le t \le j \le n$,
- на нее ссылается оператор S_{j+1}, т.е. A является аргументом или необязательным результатом S_i.

Последовательность $S_{i+1}; \ldots; S_{j+1}$ с максимальным j называется *областью действия* переменной из R'_i, если выполнены условия

- $A \notin R'_{i+1}, \ldots, R'_j$,
- $0 \le i \le t \le j \le n$.

Область влияния $S_i (y \in R'_i)$ называется *замкнутой,* если ее последним оператором S_j является такой, что $y \in R'_j$, и операторы, входящие в область влияния S_i, не содержат неявных вхождений y. Участок без замкнутых областей влияния называется *открытым.*

Оператор S_k *непосредственно зависит* от $S_i (i < k)$, если

- S_k содержится в области влияния S_i по z и $z \in A_k$;
- S_k содержится в области влияния S_i по z и $z \in R_k$;
- S_i и S_k содержатся в области влияния $S_j (j < i)$ по z и $z \in A_i \cap R_k$.

В 1-м и во 2-м случаях, при условии, что $z \notin R'_k$, зависимость является информационной, а в последнем обусловлена тем, что неявное использование требует конкретной переменной и эта переменная не может быть заменена на другую с тем же значением. Транзитивное замыкание отношения непосредственной зависимости дает отношение зависимости.

Часть неинформационных зависимостей может быть удалена с помощью следующего преобразования.

Контекст. Оператор S_i в программе таков, что область влияния по его выходной переменной x замкнута.

Преобразование 1. *Выбирается новая промежуточная переменная w, она сопоставляется выходу S_i, и все вхождения y в области*

влияния S_i заменяются на w.

Преобразование 2 (удаление бесполезных присваиваний). *Если среди результатов некоторого оператора нет активных переменных, тогда его можно удалить. Или если оператор является пересылкой и не меняет значения выходной переменной, то его тоже можно удалить.*

Преобразование 3 (исключение избыточных вычислений). *Пусть $S_i = A \leftarrow \theta B_1 \dots B_r$, $S_j = A' \leftarrow \theta B'_1 \dots B'_r$, $i < j$ $\mathrm{Val}\, S_i = \mathrm{Val}\, S_j$ и $\theta \in \Theta - \Theta'$. Тогда вводится новая промежуточная переменная w, и S_i заменяется на пару операторов $w \leftarrow \theta B_1 \dots B_r, A \leftarrow \theta_0 w$, а S_j заменяется на пересылку $A' \leftarrow \theta_0 w$.*

Остальные преобразования касаются только удаления избыточных пересылок.

Контекст. S_i — пересылка $w \leftarrow \theta_0 x$. S_j — оператор с наименьшим номером $j \geq i$, такой что $x \in R_j$. Если выделить в P подпоследовательность $S = S_{k_1}, \dots, S_{k_l} (j < k_1 < \dots < k_l)$ такую, что S_{k_r} — либо оператор, в котором w имеет явное вхождение, либо от которого зависит такой оператор, причем в S входят все такие операторы, то все операторы из S не зависят от S_j.

Преобразование 4. *Подпоследовательность S перемещается перед S_j. Удаляется S_i, и все вхождения w в S заменяются на x.*

Контекст. S_i — пересылка $x \leftarrow \theta_0 w$. S_j — единственный оператор, который присваивает w $(i > j)$. Если выделить в P подпоследовательность $S = S_{k_1}, \dots (j < k_1 < \dots < i)$, состоящую из всех операторов между S_i и S_j таких, что либо $x \in A_{k_r} \cup R_{k_r}$, либо от него зависит такой оператор, то все операторы из S не зависят от S_j. Далее выделяется подпоследовательность $S' = S_{q_1}, \dots (q < q_1 < \dots)$ и q — минимальный номер такой, что $x \in R_q$, S_{q_i} зависит от w, либо от него зависит такой оператор, и все операторы S' не зависят от S_q.

Преобразование 5. *Последовательности S и S' перемещаются соответственно перед S_j и перед S_q. Из полученной программы удаляется пересылка S_i. В S_j выходная переменная заменяется на x, и все вхождения w в области влияния S_j заменяются на x.*

3. ОПТИМИЗАЦИЯ

Пусть входу всех пересылок сопоставлены переменные, т.е. выборы аргументов всегда перевычисляются.

С каждым линейным участком связывается граф зависимостей, являющийся расширением информационного графа. Дуги этого графа, определяющие отношение непосредственной зависимости, помечаются именем переменной, а также характером зависимости: информационная или нет.

Лемма 1. *Если задан бесконтурный граф, который размечен таким образом, то можно построить его укладку с сохранением имен переменных, по которым осуществляется зависимость.*

◀ надо только заметить, что обычная укладка удовлетворяет данному свойству. ▶

В [5] введено понятие тупиковости применения преобразования T_1 по отношению к применению T_2.

Если к участку применимы $T_1(S_j)$ и $T_2(S_i)$, а после того как $T_1(S_j)$ проделано, $T_2(S_i)$ уже не применимо, будем говорить, что T_1 *тупиково* к T_2. Далее рассматривается влияние применения всех преобразований на построенный граф. Добавленные в результате применения преобразования дуги назовем *существенными,* если они создают путь между вершинами, которые до преобразования не зависели одна от другой. Существенные дуги могут определять только неинформационную зависимость.

Утверждение 1. *Преобразование T1 удаляет неинформационные зависимости. Поэтому оно не тупиково к T2, T3, T4 и T5. Значит, можно считать, что участок открытый, и после каждого T2, T3, T4 или T5 участок преобразуется к открытому.*

Отсюда, в частности, следует, что единственность оператора, присваивающего w в преобразовании T5, несущественна. Этого всегда можно добиться с помощью T1 (у промежуточных переменных нет неявных вхождений). Можно также считать, что промежуточным переменным присваивается значение только один раз.

Утверждение 2. *Преобразование T2 удаляет вершины и дуги, которые инцидентны этим вершинам. Значит, далее можно считать, что сначала удаляются все бесполезные вычисления.*

◄ Первая часть утверждения очевидна. Для доказательства второй части следует заметить, что оптимальность участка определяется числом вершин графа, а применимость всех преобразований — дугами. ►

Утверждение 3. *Преобразование T3, примененное к S_i и S_j, не удаляет операторы и не добавляет существенные дуги, если аргументы, не входящие в выборы результатов, вычисляются одними и теми же операторами.*

Утверждение 4. *После T4 могут возникнуть неинформационные зависимости. Оператор, который зависел от пересылки (т.е. присваивал значение u), будет зависеть от всех операторов, использующих w. Других существенных дуг не будет.*

◄ Зависимость операторов определяется только аргументами и результатами. Значит, утверждение следует из разбора случаев непосредственной зависимости операторов. ►

Утверждение 5. *Пусть в контексте преобразования T5 пересылка неинформационно зависит от использования или присваивания переменной u, тогда от него будет зависеть оператор S_j. От операторов, использовавших w, будут теперь неинформационно зависеть операторы, которые до преобразования неинформационно зависели от пересылки.*

◄ Полностью аналогично предыдущему. ►

Утверждение 6. *Преобразования T4 и T5 ничего не добавляют, если обе переменные промежуточные, и такую пересылку можно удалить в любой момент с помощью T4 или T5. Результат от применения T4 такой же, с точностью до переименования, как и от применения T5.*

◄ Здесь используется тот факт, что с помощью T1 участок приводится к эквивалентному, в котором присваивание промежуточной переменной делается только один раз. ►

Далее рассматриваются максимальные по включению последовательности преобразований.

В процессе применения преобразований может происходить добавление операторов (только в результате преобразования T3), а также перестановка операторов. Не уменьшая общности, можно предполагать,

что операторы пронумерованы таким образом, что оператор, информационно зависящий от другого, имеет больший номер. Это возможно, так как ни одно преобразование не меняет информационных зависимостей.

Лемма 2. *Пусть дана последовательность*

$$T_1, T_2, \ldots, T_n, \tag{1}$$

которая преобразует $\mathcal{B} = (P, V, U)$ *в* $\mathcal{B}' = (P', V, U)$, $\Theta' = \{\theta_0\}$ *и* T_i — *преобразование Т3, примененное к паре операторов с наименьшей парой, относительно следующего упорядочения (лексикографического):*

$$(i, j) < (i', j'), \text{ если } i < i' \text{ или } i = i', j < j'.$$

Тогда если $\mathcal{B} \xrightarrow{T_i} \mathcal{B}''$, *то среди всевозможных участков, которые могут быть получены из* $\mathcal{B}''$, *есть* $\mathcal{B}'$.

◀ Пусть T_i применяется к S_i и S_j $(i < j)$, тогда можно утверждать следующее: аргументы левой части S_j получаются из аргументов S_j с помощью пересылок, и их число совпадает, так как выборы аргументов всегда перевычисляются. В противном случае можно было бы найти пару с меньшим номером. Значит, как и в утверждении Т3, все пути, по которым осуществляется зависимость в S_j, могут только укоротиться:

$$T5T_i = T_iT5$$
$$T4T_i = T_iT4$$
$$T3T_i = T_iT3.$$

Из этих равенств следует утверждение о выносе в начало преобразования T_i. ▶

Замечание 1. *Если считать, что выборы аргументов не перевычисляются или есть легко вычислимая функция, кроме пересылки, то предыдущая лемма уже не будет верна, так как Т3 может добавлять существенные дуги.*

Теперь можно считать, что вначале применяются операторы Т3 с убывающими номерами пар операторов, к которым они применяются. После того как применили все Т3, получится следующая ситуация: левая часть всех операторов, кроме пересылок, вычисляет уникальное значение, которое потом с помощью нескольких пересылок передается в те операторы $\mathcal{B}$, которые их используют.

Теорема 3. *Если* $\Theta' = \emptyset$ *или* $\Theta' = \{\theta_0\}$, *оптимизацию можно сделать так:*

- *сначала применяются всевозможные T2, как указано выше;*
- *потом все T3;*
- *наконец, все T4 и T5.*

Удаление тривиальных пересылок, по утверждению 6, можно делать на любом шаге.

◀ Следует из утверждений 2 и 6 и леммы 2. ▶

Теорема 4. *Задачу о размыкании всех контуров ориентированного графа* G *можно за полиномиальное время свести к задаче об оптимизации участка, в котором имеются неявные вхождения переменных, и есть, по крайней мере, одна операция, отличная от пересылки.*

◀ Каждой вершине графа G будет соответствовать вычисление разных выходных переменных из разных входных, причем эти вычисления вначале не связаны. Каждой дуге графа G будет соответствовать пересылка, к которой применимо T4, эта пересылка при удалении создает только одну существенную дугу. Эта дуга соответствует дуге данного графа. В некоторых случаях потребуется, чтобы число аргументов оператора было больше 1, это достигается использованием выборов аргументов.

Построение проводится постепенно, добавлением одной дуги за раз. Добавление дуги (a, b) происходит следующим образом. Если уже построены дуги, соответствующие дугам (x, a), то новая дуга начинается в операторе, достижимом из всех операторов, являющихся концами существенных дуг, соответствующих построенным дугам (x, a). Аналогично, операторы начала дуг (b, y), должны быть достижимы из конца новой существенной дуги.

Несложно заметить, что оптимизация свелась к удалению пересылок, а удаление пересылок эквивалентно применению максимального числа преобразований T4, что, в свою очередь, эквивалентно удалению минимального количества ребер графа G для размыкания всех контуров. Контекст T3 не выполняется никогда, так как выходные переменные зависят от разных входных переменных. Для каждой пересылки $w \leftarrow \theta_0 v$ требуется одна входная переменная v, одно использование промежуточной переменной w, одно неявное использование v, после одного присваивания v. Отсюда следует утверждение. ▶

Теорема 5. *В общем случае (не налагаются никакие условия на Θ' и на использование выбора результата) оптимизация делается за полиномиальное время недетерминированным алгоритмом.*

◀ Длина цепочки преобразований, которые могут быть применены к участку, ограничена числом $4n$, где n — длина программы. Два раза к одному оператору можно применить различные преобразования, только если первое было Т3. Добавляются операторы только с помощью Т3, и число новых операторов не превышает n. ▶

Следствие 1. *Задача оптимизации NP-полная.*

4. МОДЕЛЬ С. МАТВИНА

Лемма 6. *У открытых участков графы изоморфны с сохранением разметки тогда и только тогда, когда любой из них можно преобразовать в другой при помощи перестановки соседних независимых операторов и переименования промежуточных переменных.*

◀ Достаточность очевидна. Требуется доказать необходимость. Для каждой перестановки $i_1 \dots i_n$ чисел $1 \dots n$ вводится четность — число пар чисел от $(i_k, i_j), 1 \le k < j \le n$ и при этом $i_k > i_j$. Теперь операторы первого блока нумеруются, и рассматривается перестановка, соответствующая второму блоку. Если четность равна 0, то после изменения имен промежуточных переменных получаются одинаковые участки. Если четность не равна 0, то во втором участке есть стоящие рядом операторы с номерами i и j, причем первым стоит оператор с большим номером. Эти операторы независимы вследствие изоморфизма графов, значит, их можно переставить и уменьшить четность на единицу. Такие пары операторов ищутся следующим образом: выделяются группы операторов с возрастающими номерами операторов и берутся пограничные операторы. Они всегда есть, если число групп больше одной, что эквивалентно тому, что четность отлична от нуля. ▶

Следствие 2. *Если допустить перестановку соседних независимых операторов, то в схеме оптимизации в теореме 3 ничто не изменится.*

Если в участке нет выборов результатов и аргументов, преобразования можно применять в любую сторону и $\Theta' = \{\theta_0\}$, то получается модель, введенная Матвиным в [6]. Далее рассматривается эта модель.

Предполагается известным тот факт, что любой участок, эквивалентный данному в смысле определения, можно преобразовать к данному с помощью преобразований Т2 и Т3. Доказательство приводится, например, в [3].

Утверждение 7. *Можно считать, что каждая переменная появляется в левой части присваивания не более одного раза.*
◀ Это достигается переименованием переменных, которое является комбинацией преобразований Т2 и Т3. ▶

Теперь по участку строится граф G. Вначале рассматривается информационный граф G' выражений, в нем сразу же удаляются вершины, не используемые для вычисления значений выходных переменных. Далее каждой вершине этого графа приписываются имена переменных, в которых должно оказаться соответствующее выражение (входной вершине приписывается также соответствующая входная переменная). Требуется дополнить граф G' неинформационными зависимостями, аналогичными зависимостям из определения 2. Эти зависимости вследствие утверждения 7 могут появиться только для переменных из $U \cap V$. Переменные из $U \cap V$ называются далее *транзитными.* Точнее, оператор, которому приписана транзитная переменная, зависит неинформационнно от каждого оператора, использующего входное значение этой переменной, всем таким дугам соответствует данная переменная. Петли не добавляются. В итоге получается граф G.

Каждой вершине соответствует одно вычисление (либо оператор **старт**, если вершине отвечает входная переменная) и $n-1$ пересылка, где n — число переменных, приписанных вершине.

Так как граф зависимостей по аргументам бесконтурный, то в любом контуре графа G есть дуга неинформационной зависимости.

Теперь сформулируем критерий, когда можно построить по графу G участок. Пусть дана укладка графа G. Дуга неинформационной зависимости называется *фиктивной,* если оператор ее начала не использует (у входной переменной есть копия) или оператор ее конца не присваивает соответстсвующей переменной. Пусть задан граф G, удовлетворяющий условиям:

- каждой дуге приписано r или a (r соответствует неинформационной зависимости, a — информационной);
- каждая вершина помечена либо как операция $\theta \in \Theta$, либо как входная переменная;

- зафиксирован порядок a-дуг, входящих в каждую вершину;
- граф G', получающийся удалением всех r-дуг из G, бесконтурный;
- каждой вершине, помеченной операцией $\theta \in \Theta$, приписан список переменных, причем каждая переменная может появляться в нем не более одного раза и только, если она промежуточная или выходная;
- r-дуги определены так же, как и ранее, и им приписаны имена транзитных переменных.

Утверждение 8. *По графу G, описанному выше, можно построить участок, если, и только если, в каждой вершине можно выбрать приписанную переменную так, чтобы в любом контуре имелась фиктивная r-дуга.*

◀ Достаточность. Очевидный факт, так как иначе r-дуги накладывают ограничение на порядок расположения операторов.

Необходимость. Удалив фиктивные дуги, получаем бесконтурный граф, по которому строится участок. Выбранной переменной присваивается значение каждой вершины, которое пересылается в остальные переменные в конце. Для входных вершин это правило изменяется так: если входной вершине приписана промежуточная переменная, то в ней вначале сохраняется значение входной переменной для последующего использования. Корректность проверяется тривиально. ▶

Теперь можно оптимизировать участок. Сначала рассматриваются все операторы, которые нельзя исключить. Для этого в каждой вершине оставляются только выходные переменные, если таких нет, то единственная промежуточная переменная, приписанная вершине, остается (получается граф G_{st}). Если при такой разметке можно выбрать переменые, как в утреждении 8, то полученный участок будет оптимальным. Если нет, то требуется приписать минимальное количество новых промежуточных переменных некоторым вершинам, чтобы это стало возможным.

Теорема 7. *Построенный таким образом участок будет оптимальным относительно обоих критериев, указанных в [6].*

◀ Сравним граф оптимального относительно любого критерия участка с построенным выше. Они будут содержать одинаковое число вершин. Значит, осталось минимизировать только число пересылок, но

в построенном участке оно минимально. Следовательно, построенный участок оптимальный относительно обоих критериев. ►

Утверждение 9.

1) Если в некоторой вершине a графа G_{st} есть нетранзитная переменная u, то можно считать, что значение вершины присваивается u, поэтому r-дуги, входящие в нее, всегда рассматриваются как фиктивные и удаляются.

2) Если через некоторую r-дугу D не проходит ни один контур, то все r-дуги, входящие в ее конец a, можно удалить, так как выходной переменной оператора вершины a будет переменная, соответствующая дуге D.

◄ Просто требуется заметить, что если присваивать, как указано в условиях, то не добавится контуров без фиктивных r-дуг. ►

Лемма 8. *Если в графе G оптимизированного участка некоторой входной вершине v приписана промежуточная переменная, то, удалив v и приписав новую переменную вершине θ, соответствующей выходному значению v, получится граф некоторого другого оптимального участка.*

◄ Приписыванием промежуточной переменной входной вершине v получается, что можно считать фиктивными только r-дуги, соответствующие v. Если же добавить новую переменную в вершину θ, то фиктивными станут еще и r-дуги, соответствующие другим переменным, приписанным θ. ►

Далее рассматривается вспомогательный мультиграф H. Его вершинами будут все невходные вершины графа G_{st}, не удовлетворяющие условиям утверждения 9. Две вершины t и z графа H соединяются дугой (t, z), если существует путь из t в z в графе G_{st}, состоящий из a-дуг и r-дуг, соответствующих некоторой фиксированной переменной u, приписанной z. Эта дуга помечается именем u. В мультиграфе H могут быть петли.

В соответствии с утверждением 8 требуется введение новых переменных, для того чтобы в графе G_{st} было достаточно фиктивных дуг для размыкания всех контуров. Введение новых переменных моделируется упрощениями мультиграфа H.

1) Выбирается и удаляется некоторая вершина и все ребра, инцидентные ей, в H (это соответствует приписыванию соответствующей вершине графа G_{st} новой промежуточной переменной).
2) Если появились вершины t без входящих ребер, помеченных некоторой переменной, приписанной t, то t удаляется вместе со всеми инцидентными ребрами. Это проделывается пока возможно.

Теорема 9. *Задача оптимизации сводится к минимизации числа упрощений для получения пустого графа.*

◀ По утверждению 9 и лемме 8 новые промежуточные переменные добавляются только в вершины графа H. Если мультиграф H станет пустым, то по графу G_{st} можно построить участок. С каждой вершиной H, а с ней и вершиной графа G_{st} связывается либо новая переменная, либо переменная, для которой не было входящих дуг во время выполнения какой-нибудь очередной операции (см. утверждение 9).

Если мультиграф H после некоторого шага не пуст, то по графу G_{st} невозможно построить участок. Доказательство проводится от противного. Рассматриваются все операторы, которые соответствуют вершинам H. Для каждого из них есть предшествующий, так как есть входные дуги мультиграфа H, соотвествующие всем переменным, приписанным вершине графа G_{st}. В этом состоит противоречие.

Число операций над мультиграфом H равно количеству пересылок, добавленных для того, чтобы не осталось контуров графа G_{st} с фиктивными дугами. ▶

Следствие 3. *Если в графе G_{st} каждой вершине приписана только одна переменная, то задача оптимизации сводится к поиску минимального числа вершин графа H (в нем не будет кратных ребер), размыкающих все контуры H. Задача оптимизации NP-полная.*

Утверждение 10. *Построение мультиграфа H можно проделать за время $O(n^3)$, где n — длина программы.*

◀ Построение графа G_{st} делается за время $O(n^2)$, а по графу G_{st} мультиграф H строится за время $O(n^3)$. ▶

Очевидно, что можно построить недетерминированный алгоритм оптимизации. Отсюда следует NP-полнота.

5. ЗАКЛЮЧЕНИЕ

В работе показано, что оптимизация линейных участков в некоторых случаях не может быть выполнена за полиномиальное время, и поэтому требуется формулировка условий, при выполнении которых возможна полная оптимизация. Показано, что для участка С. Матвина сложность определяется числом транзитных переменных, а в участке И. В. Поттосина — числом пересылок в исходном участке.

Приближенный алгоритм задачи оптимизации можно строить по приближенному алгоритму решения соответствующей задачи теории графов, так как построенные алгоритмы сведения не приводят к увеличению абсолютной погрешности приближенного решения задачи оптимизации по сравнению с погрешностью решения задачи теории графов.

СПИСОК ЛИТЕРАТУРЫ

1. **Ахо А., Ульман Дж.** Теория синтаксического анализа, перевода и компиляции: Пер. с англ. — М.: Мир, 1978. — Т. 2.
2. **Касьянов, В. Н.** Оптимизирующие преобразования программ. — М.: Наука, 1988. — 336 с.
3. **Касьянов, В. Н.** Эквивалентные преобразования линейных участков программ // Трансляция и преобразования программ. — Новосибирск, 1984. — С. 56–61.
4. **Поттосин, И. В.** Направленные преобразования линейного участка // Языки и системы программирования. — Новосибирск, 1981. — 17 с.
5. **Поттосин, И. В.** Об алгоритме оптимизации линейных участков // Трансляция и преобразования программ. — Новосибирск, 1984. — 14 с.
6. **Matwin S.** On the completeness of a set of transformations optimizing linear programs // Inf. proces. letters. — 1977. — Vol. 6. — P. 165–167