**ИНФОРМАТИКА**

**2007** **январь-март** **№ 1**

*МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ*

*УДК 519.711.3*

**А.С. Поляков, В.Е. Самсонов**

# МИНИМИЗАЦИЯ ЗАТРАТ НА ПОСТРОЕНИЕ МОДЕЛЕЙ ПОСЛЕДОВАТЕЛЬНЫХ ПРОГРАММ ПРИ ИХ РАСПРЕДЕЛЕННОЙ РЕАЛИЗАЦИИ

*Рассматривается задача минимизации затрат на построение математических моделей последовательных программ при их распределенной реализации. В качестве моделей программ используются ориентированные графы, веса вершин которых представляют собой числа выполнений линейных участков, веса дуг – числа переходов между линейными участками при решении задач с помощью рассматриваемой программы. Предлагаются алгоритмы поиска минимальных по мощности множеств вершин и дуг графа, задание значений весов которых позволяет вычислить веса всех остальных вершин и дуг графа.*

## Введение

При решении многих задач используются математические модели различных объектов в виде ориентированных графов, вершинам и дугам которых «приписываются» веса, характеризующие свойства рассматриваемого объекта. Такого рода графы применяются при определении временных характеристик программ [1], оптимизации программ с помощью метода динамического использования регистров [2], распределенной реализации программ [3, 4] и др. В указанных случаях вершины графа соответствуют линейным участкам программы, а дуги – переходам между ними. Веса вершин и дуг графа представляют собой вычислительную сложность линейных участков программы и информационную сложность связей между ними соответственно.

Для вычисления этих весов необходимо знать числа выполнений линейных участков и переходов между ними при реализации программы. Для нахождения значений таких величин существуют различные способы, использующие, в частности, модели программ, основанные на предположении, что вероятности переходов в графе программ постоянны, а операторы не влияют на значения логических условий. Однако, как показано в работах [5–7], такие модели не позволяют получить достаточно точные характеристики программ.

Точный способ получения значений указанных характеристик программ состоит в непосредственном измерении чисел выполнения линейных участков и переходов между ними при решении с помощью рассматриваемой программы некоторой «типичной» задачи либо достаточно большой серии обычных задач. Поскольку для получения требуемых данных в текст исходной программы вставляются счетчики, это приводит к увеличению размера программы и времени ее выполнения. Для сокращения затрат необходимо разработать способ нахождения минимального множества линейных участков и переходов, на которых обязательна установка счетчиков, а также алгоритм вычисления по показаниям этих счетчиков чисел выполнения остальных линейных участков и переходов.

## 1. Модель программы. Постановка задачи

В качестве модели программы используем ориентированный граф $G = (X,U)$, где $X$ и $U$ – множества вершин и дуг, представляющие соответственно множества линейных участков программы и переходов между ними. Учитывая свойства реальных программ, можно заметить, что любая вершина $x_i$ имеет непустые множества (классы) инцидентных ей заходящих (обозначим их через $U_i^+$) и исходящих ($U_i^-$) дуг. Каждой вершине $x_i \in X$ и дуге $u_{jk} \in U$ зададим веса $v_i > 0$ и

$w_{jk} > 0$ соответственно, представляющие собой числа выполнений линейных участков и переходов между ними при выполнении программы.

Для эффективного решения задачи распределенной реализации последовательной программы необходимо знать числа выполнений всех линейных участков и переходов. Поскольку установка счетчиков приводит к повышению вычислительных затрат, увеличению объема программы и времени ее выполнения, число устанавливаемых счетчиков нужно минимизировать. Разработка алгоритма нахождения минимального множества вершин и дуг графа, по значениям весов которых можно вычислить веса остальных вершин и дуг, требует предварительного решения следующих задач.

*Задача 1*. По заданным весам всех вершин графа найти *сбалансированное распределение* весов вершин по дугам, т. е. вычислить веса дуг графа таким образом, чтобы для любой вершины $x_i \in X$ выполнялось условие

$$w(U_i^+) = w(U_i^-) = v_i, \tag{1}$$

где $w(U_i^+)$ и $w(U_i^-)$ – суммы весов дуг множеств $U_i^+$ и $U_i^-$ соответственно. Для графов линейных участков реальных программ, веса вершин которых определяются с помощью счетчиков в процессе выполнения программы, существование такого распределения очевидно.

*Задача 2.* Найти минимальное множество вершин и дуг графа, по известным значениям весов которых можно вычислить веса всех вершин и дуг графа. Разработать соответствующий алгоритм вычисления весов вершин и дуг.

## 2. Нахождение сбалансированного распределения весов вершин по дугам

Введем следующие определения. *Чередующейся цепью* в ориентированном графе назовем цепь, у которой все дуги различны и любые две смежные дуги имеют противоположное направление, а *чередующимся циклом* – чередующуюся цепь, у которой первая и последняя дуги инцидентны одной и той же вершине графа. *Определяющее множество S* – множество дуг, в котором есть, по крайней мере, по одной дуге из каждого чередующегося цикла графа, *минимальное определяющее множество* $S_{min}$ – определяющее множество минимальной мощности. Алгоритм поиска сбалансированного распределения весов вершин по дугам графа (задача 1) основан на следующих утверждениях.

*Утверждение 1*. Необходимым и достаточным условием единственности сбалансированного распределения весов вершин по дугам является отсутствие в графе чередующихся циклов.

Доказательство необходимости. Предположим противное: пусть в графе с единственным сбалансированным распределением весов вершин по дугам имеется чередующийся цикл $C$. Среди дуг этого цикла выберем дугу $u_{ij}$ с наименьшим весом $w_{ij}$ и увеличим ее вес на величину $0 < \Delta w < w_{ij}$. Увеличим также на $\Delta w$ веса всех дуг цикла $C$, имеющих такое же направление, как и дуга $u_{ij}$, и уменьшим на $\Delta w$ веса всех дуг, имеющих противоположное направление. Очевидно, что при этом ни у одной из дуг цикла $C$ вес не станет равным или меньшим 0, а суммы весов дуг классов $U_i^+$ и $U_i^-$ для любой из вершин цикла не изменятся. Следовательно, полученное распределение является сбалансированным, но отличающимся от исходного, что противоречит исходному предположению.

Доказательство достаточности. Пусть в графе без чередующихся циклов имеется некоторое сбалансированное распределение весов вершин по дугам. Предположим, что оно не единственное и существует некоторое другое сбалансированное распределение. Допустим, что вес некоторой дуги $u_{ij} \in (U_i^- \cap U_j^+)$ в этом распределении на $\Delta v$ больше, чем в исходном (если вес дуги $u_{ij}$ в новом распределении меньше, чем в исходном, доказательство проводится аналогично). Чтобы новое распределение было сбалансированным, необходимо суммарный вес остальных дуг класса $U_j^+$ уменьшить на $\Delta v$. Допустим, что изменения весов дуг этого класса равны $\Delta w_{1j}, \Delta w_{2j}, ..., \Delta w_{kj}$, $u_{ij} \in U_j^+$, $i \in \{1, ..., k\}$. Выберем одну из этих дуг, например $u_{mj} \in (U_m^- \cap U_j^+)$, и рассмотрим вершину $x_m$. Для нее аналогично должны измениться веса дуг класса $U_m^-$. Поскольку распределение весов вершин по дугам, согласно исходному предполо-

жению, является сбалансированным, а число вершин в графе конечно, то продвижение по чередующейся цепи должно привести в одну из уже пройденных вершин (например, $x_p$). Если последняя дуга $u_{пр}$ данной цепи принадлежит к тому же классу дуг, что и дуга, по которой осуществлялось прохождение через вершину $x_p$ в предыдущий раз, это свидетельствует о наличии в графе чередующегося цикла, что противоречит исходному предположению.

Если же дуга $u_{пр}$ принадлежит противоположному классу дуг, то можно осуществить продвижение по чередующейся цепи дальше. Такое продвижение неизбежно должно закончиться в одной из пройденных ранее вершин, при этом последняя дуга чередующейся цепи должна принадлежать к тому же классу, что и дуга, по которой осуществлялось прохождение через эту вершину ранее (иначе в данной вершине не будет равенства веса вершины и суммарного веса дуг одного класса). В таком случае в графе существует чередующийся цикл, что также противоречит исходному предположению об отсутствии таковых.

*Утверждение 2*. В графе без чередующихся циклов имеется, по крайней мере, одна вершина $x_j$, для которой выполняется условие

$$(|U_j^+| = 1) \vee (|U_j^-| = 1), \tag{2}$$

где $|M|$ – мощность множества $M$.

Доказательство. Возьмем произвольную вершину $x_i \in X$ и будем двигаться из нее по чередующемуся пути, начиная с дуги того класса, мощность которого минимальна. При этом возможны следующие исходы: мощность выбранного класса дуг вершины $x_i$ равна 1 и доказательство закончено; мощность выбранного класса дуг больше 1. В этом случае при движении из вершины $x_i$ по чередующейся цепи возможны следующие ситуации.

1. Очередная вершина относится к множеству еще не пройденных вершин и у нее нет другой дуги того же класса, что и у последней из пройденных дуг чередующейся цепи. Следовательно, мощность данного класса дуг для этой вершины равна 1, что и требовалось доказать.

2. Очередная встреченная вершина уже однажды была пройдена. Поскольку в графе нет чередующихся циклов, то последняя дуга пройденной цепи принадлежит другому классу дуг, чем та дуга, по которой эта вершина была пройдена в предыдущий раз. Дальнейшие исходы таковы: из встреченной вершины выйти нельзя, следовательно, мощность класса, которому принадлежит последняя из пройденных дуг чередующейся цепи, равна 1 и доказательство закончено; из вершины возможно продвижение дальше. Попасть в одну и ту же вершину в третий раз нельзя, поскольку это будет означать, что в графе имеется чередующийся цикл (поскольку предыдущие два раза прохождение через вершину осуществлялось по дугам разных классов). Поскольку множество вершин конечно, продвижение по чередующейся цепи должно закончиться в вершине, из которой нет второй дуги того класса, к которому принадлежит последняя из пройденных дуг.

*Утверждение 3*. В графе с известными значениями весов всех вершин и дуг любого определяющего множества существует единственное сбалансированное распределение весов вершин по дугам.

Доказательство. Дуги определяющего множества с известными весами удалим из графа, в результате получим граф $G' = (X,U')$ без чередующихся циклов. В соответствии с утверждением 2 в $G'$ имеется вершина $x_i$, для которой справедливо равенство $|U_i'^{\sim}| = 1$, $\sim \in \{+, -\}$. Вес единственной дуги $u \in U_i'^{\sim}$ легко вычисляется по формуле

$$w(u) = v_i - \sum w(\check{U}_i'^{\sim}), \tag{3}$$

где $v_i$ – вес вершины $x_i$; $\check{U}_i'^{\sim}$ – множество дуг, веса которых известны. Дугу $u$ удалим из $G'$. Поскольку при этом в графе по-прежнему нет чередующихся циклов, то можно найти следующую вершину, удовлетворяющую условию (2), и вычислить вес одной из инцидентных ей дуг. Поступая таким образом, можно вычислить веса всех дуг графа. Полученное распределение весов вершин по дугам будет сбалансированным в связи с тем, что вычисление весов дуг производи-

лось по формуле (3), полученной из (1), и единственным в соответствии с утверждением 1, поскольку в графе нет чередующихся циклов.

### 3. Алгоритм распределения весов вершин по дугам

Используя доказанные выше утверждения 1–3, можно предложить алгоритм сбалансированного распределения весов вершин по дугам графа, состоящий из указанных ниже операций.

1. В графе $G$ отыскиваются вершины, удовлетворяющие условию (2), и удаляются инцидентные им дуги, единственные в своем классе, а также изолированные вершины, которые могут образоваться вследствие выполнения указанных операций. Если в результате будет получен пустой подграф, то производится переход в п. 2. В противном случае в оставшемся частичном подграфе $G_p = (X_p, U_p)$ отыскивается множество $K$ всех чередующихся циклов и находится минимальное определяющее множество $S_{min}$, представляющее собой кратчайшее столбцовое покрытие булевой матрицы, строки которой соответствуют чередующимся циклам, а столбцы – дугам чередующихся циклов.

2. Программа, моделью которой является рассматриваемый граф, расширяется путем установки счетчиков на всех линейных участках и переходах, соответствующих дугам множества $S_{min}$. С помощью полученной программы производится решение некоторой «типичной» задачи, в результате чего определяются веса всех вершин графа и дуг множества $S_{min}$.

3. В исходном графе $G$ с известными весами всех вершин и дуг множества $S_{min}$ отыскиваются поочередно вершины $x_i$, для которых выполняется условие:

$$\exists\, U_i^{\sim} \,/\, |U_i^{\sim} \setminus \check{U}_i^{\sim}| = 1,\ \sim \in \{+, -\}. \tag{4}$$

Вес единственной неизвестной дуги $u \in U_i^{\sim}$ вычисляется по формуле (3). Алгоритм заканчивает работу, когда будут найдены веса всех дуг графа.

Рассмотрим работу алгоритма на примере графа, показанного на рис. 1.

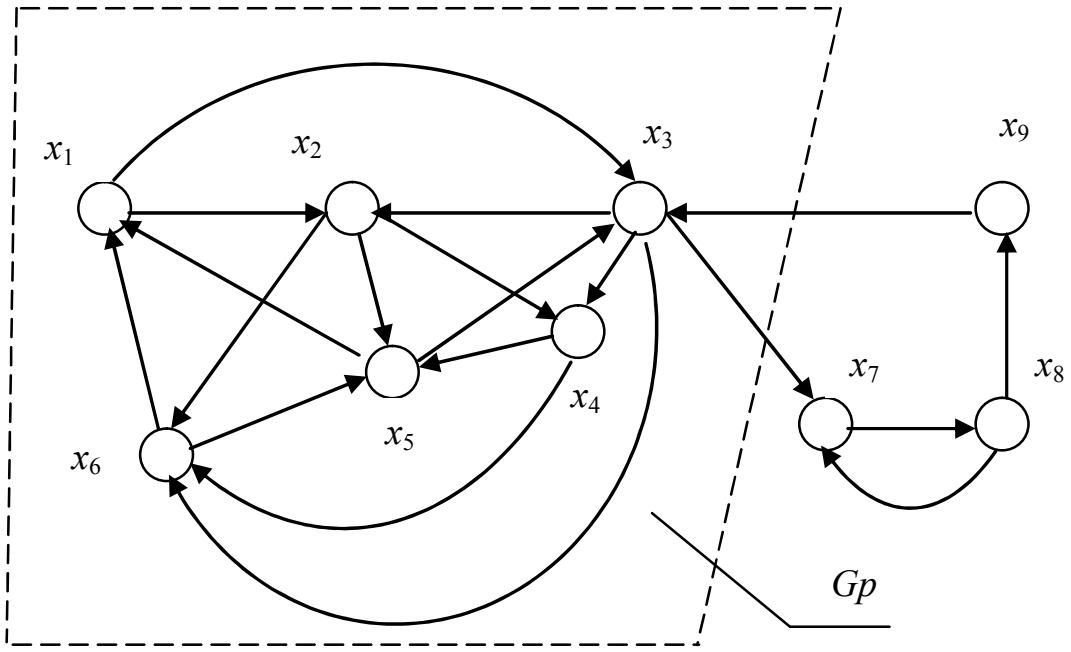


Рис. 1. Пример графа линейных участков (*Gp* – частичный подграф)

1. Поочередно находим вершины $x_7$, $x_9$, $x_8$, $x_7$, удовлетворяющие условию (2), и удаляем дуги $(x_7, x_8)$, $(x_9, x_3)$, $(x_8, x_9)$, $(x_8, x_7)$, $(x_3, x_7)$, а затем и вершины $x_7$, $x_8$, $x_9$, поскольку они оказались изолированными. Получаем подграф $G_p$, отыскиваем в нем множество $K$ чередующихся циклов

и строим булеву матрицу, в которой находим кратчайшее столбцовое покрытие, состоящее из столбцов $(x_2, x_4)$, $(x_2, x_5)$, $(x_1, x_2)$:

$U_p$

| | $(x_1, x_2)$ | $(x_1, x_3)$ | $(x_2, x_4)$ | $(x_2, x_5)$ | $(x_2, x_6)$ | $(x_3, x_2)$ | $(x_3, x_4)$ | $(x_3, x_6)$ | $(x_4, x_5)$ | $(x_4, x_6)$ | $(x_5, x_1)$ | $(x_5, x_3)$ | $(x_6, x_1)$ | $(x_6, x_5)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k_1$ | | | 1 | | 1 | | 1 | 1 | | | | | | |
| $k_2$ | | | 1 | 1 | | | 1 | 1 | 1 | 1 | | | | |
| $k_3$ | | | | 1 | 1 | | | | 1 | 1 | | | | |
| $k_4$ | 1 | 1 | | | | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $k_5$ | 1 | 1 | 1 | 1 | | 1 | 1 | | | | 1 | 1 | 1 | 1 |
| $k_6$ | 1 | 1 | | 1 | 1 | 1 | | 1 | | | 1 | 1 | 1 | 1 |
| $k_7$ | 1 | 1 | 1 | | 1 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | 1 |

Соответствующие дуги графа образуют минимальное определяющее множество $S_{min}$ = $= \{(x_2, x_4), (x_2, x_5), (x_1, x_2)\}$.

2. Веса вершин графа, а также дуг множества $S_{min}$ изображены на рис. 2 (дуги множества $S_{min}$ показаны штриховыми линиями).

3. В исходном графе с известными весами вершин и дуг множества $S_{min}$ поочередно находим вершины, для которых выполняется условие (4), и вычисляем веса соответствующих дуг (на рис. 2 очередность вычислений показана с помощью чисел 1–16).

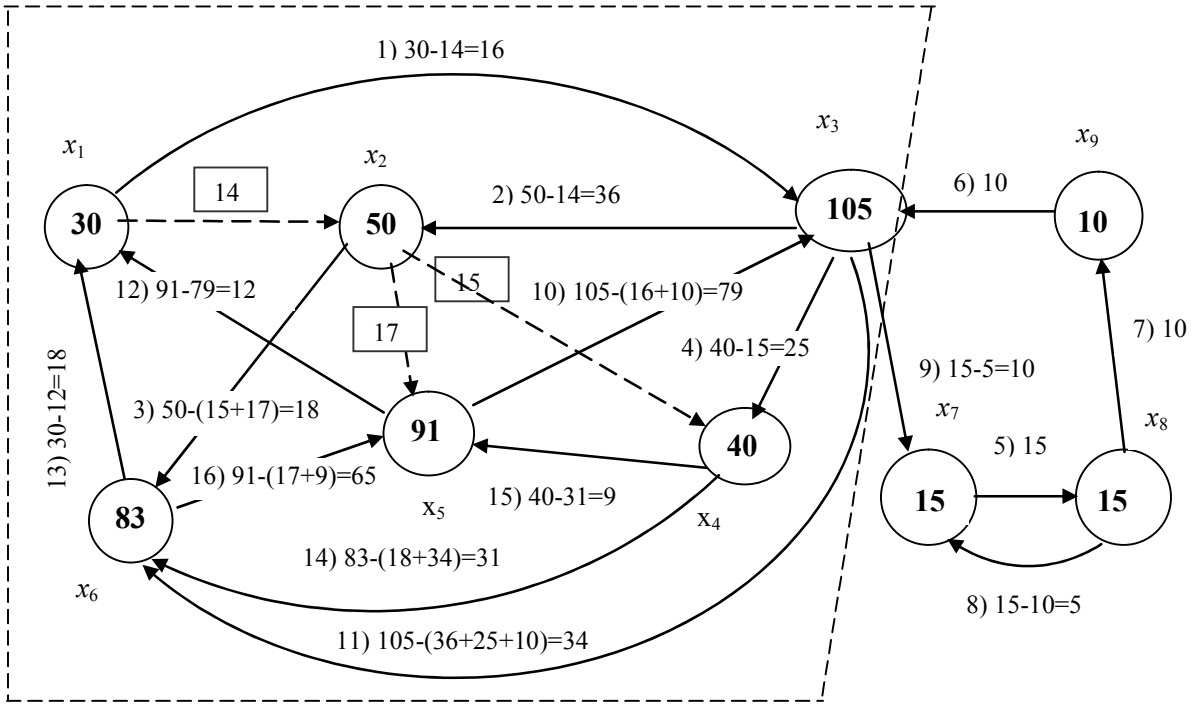


Рис. 2. Последовательность вычисления весов дуг

### 4. Минимизация числа счетчиков в программе

Введем операцию *расщепление дуг*, заключающуюся в том, что в графе G каждая дуга минимального определяющего множества $S_{min}$ заменяется вершиной с парой инцидентных ей дуг различных классов, как показано на рис. 3 для дуг $(x_1, x_2)$, $(x_2, x_4)$, $(x_2, x_5)$. Применение такой операции приводит к разрыву всех имеющихся в нем чередующихся циклов и появлению но-

вых вершин $x_k$, $x_m$, $x_r$. Преобразованный таким образом граф представляет собой граф линейных участков программы после установки счетчиков на переходах, соответствующих дугам определяющего множества $S_{min}$.

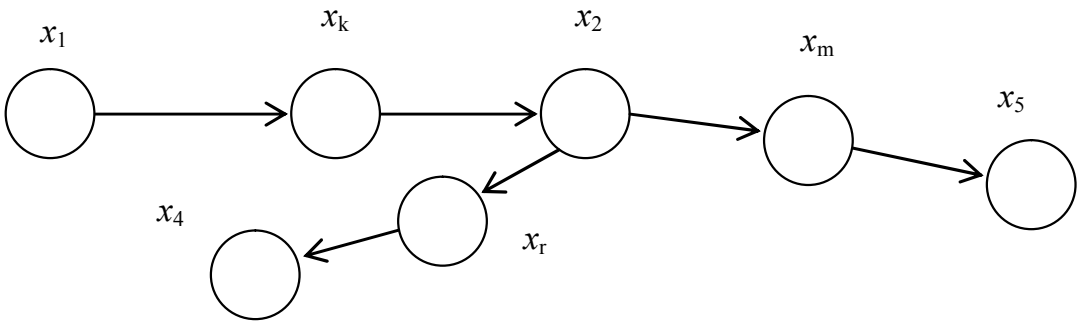


Рис. 3. Пример «расщепления» дуг минимального определяющего множества

*Независимым множеством вершин N* графа, не содержащего чередующихся циклов, назовем множество вершин, по значениям весов которых можно вычислить веса всех вершин и дуг графа путем последовательного нахождения вершин $x_i$, для которых выполняется условие (4) и известен вес вершины $x_i$, либо веса всех инцидентных ей дуг класса, противоположного тому, для которого выполняется условие (4). В первом из этих случаев легко вычислить вес единственной из дуг множества $U_i^{\sim}$, а во втором – сначала вес вершины $x_i$, а затем вес единственной из инцидентных ей дуг с неизвестным весом.

*Минимальным независимым множеством вершин* графа назовем независимое множество вершин, имеющее минимальную мощность. Процесс нахождения множества $N_{min}$, т. е. минимального числа счетчиков, устанавливаемых в программе, состоит из двух этапов. Поскольку знание весов дуг определяющего множества графа необходимо для обеспечения единственности решения, то вначале нужно найти некоторое минимальное определяющее множество $S_{min}$ и таким образом зафиксировать переходы между линейными участками, на которых обязательна установка счетчиков. Затем граф преобразуется путем применения операции расщепления дуг. В результате этого получается граф G', не имеющий чередующихся циклов, в котором отыскивается минимальное независимое множество вершин $N_{min}$.

*Утверждение 4.* Множеству $N_{min}$ принадлежат все вершины графа, имеющие петли, и, по крайней мере, по одной вершине из каждого контура длины 2.

Действительно, вес петли, инцидентной некоторой вершине, может быть вычислен только в том случае, если известен вес этой вершины, который не может быть определен через веса инцидентных ей дуг (не считая дуги, образующей петлю), поскольку петля рассматривается как пара дуг противоположных классов, веса которых неизвестны. Следовательно, вершина, имеющая петлю, должна принадлежать множеству $N_{min}$.

Аналогичная ситуация возникает и при наличии в графе контура длины 2. В этом случае веса дуг, инцидентных одновременно обеим вершинам такого контура, могут быть найдены только если известен вес, по крайней мере, одной из его вершин, при этом должны быть известны и веса всех остальных инцидентных ей дуг. Поэтому одна из вершин каждого контура длины 2 должна обязательно принадлежать множеству $N_{min}$.

### 5. Алгоритм поиска минимального независимого множества вершин

Рассмотрим алгоритм решения поставленной задачи с помощью приведенных выше утверждений 2 и 4. Поиск множества $N_{min}$ производится таким образом, чтобы уменьшить общую сумму значений счетчиков, устанавливаемых на линейных участках программы. Это достигается благодаря учету присущей реальным программам неравномерности распределения нагрузки между линейными участками. Из практики известно, что при работе программ чаще всего реализуются циклы, состоящие из небольшого числа вершин, а также линейные участки, имеющие большее число связей с другими линейными участками.

1. Образуем множество $N_t$, в которое включим вершины: появившиеся в результате применения операции расщепления дуг; имеющие петли; из каждого контура длины 2 – имеющие меньшие локальные степени.

2. Если множество вершин графа пусто, то переход в п. 5, иначе поочередно рассматриваются вершины графа, принадлежащие $N_t$. Если среди них нет вершин, удовлетворяющих условию (2), то переход в п. 3. Иначе из графа удаляются дуги, наличие которых обеспечивает выполнение условия (2) для вершин, принадлежащих $N_t$, а также удаляются изолированные вершины, которые могут при этом образоваться. Переход в п. 3.

3. Поочередно находим вершины, локальные степени которых равны 1. Если таких вершин нет, то переход в п. 4. В противном случае найденные вершины удаляются из графа и осуществляется переход в п. 2.

4. Находим вершину, для которой выполняется условие (2). Если таких вершин несколько, вначале выбираем ту, для которой длина минимального из проходящих через нее контуров является наибольшей (при этом предпочтение отдается вершинам с меньшей локальной степенью). Дуга, определяющая выполнение условия (2) для выбранной вершины, удаляется из графа, а сама вершина включается в $N_t$. Переход в п. 2.

5. Текущее значение полученного множества $N_t$ принимаем в качестве $N_{min}$.

Работу данного алгоритма рассмотрим на примере графа линейных участков программы МАКСПОДГРАФ [8], показанного на рис. 4. В этом графе отсутствуют чередующиеся циклы и вершины с петлями, поэтому в $N_t$ сначала включаются вершины $h$ и $k$, входящие в контуры длины 2 и имеющие в них меньшие локальные степени. Применение остальных пунктов алгоритма приводит к нахождению множества $N_t = \{h, k, f, i, d, c, a, n\}$. Следовательно, для рассматриваемого графа множество $N_{min} = \{a, c, d, f, h, i, k, n\}$. Элементы $N_{min}$ отмечены знаком «+», а очередность их нахождения указана цифрами 1–8.

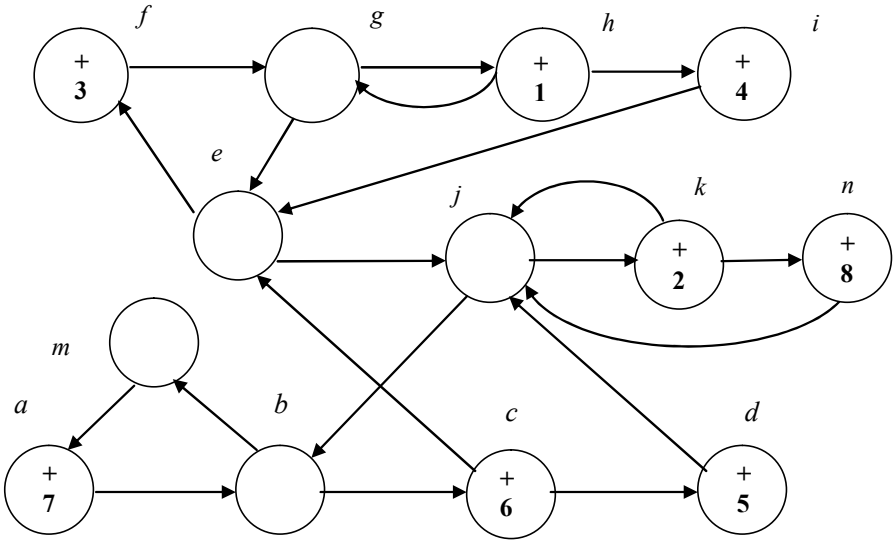


Рис. 4. Граф линейных участков программы МАКСПОДГРАФ

В графе линейных участков с известными значениями элементов $N_{min}$ легко вычислить веса вершин и дуг с помощью п. 3 алгоритма нахождения сбалансированного распределения весов вершин по дугам графа, дополненного операцией вычисления весов тех вершин, у которых образуются инцидентные дуги одного класса с известным весом.

### 6. Результаты экспериментов. Оценка вычислительной сложности алгоритмов

Вычислительная сложность операций, используемых в приведенных выше алгоритмах поиска минимального независимого множества вершин и нахождения сбалансированного распределения весов вершин по дугам, находится в прямой зависимости от количества вер-

шин и дуг в графе. Это позволяет сделать вывод о том, что алгоритмическая сложность предлагаемых алгоритмов является линейной функцией размера графа, определяемого числом вершин и дуг.

Для определения эффективности алгоритмов были проведены вычислительные эксперименты с применением программ поиска множества максимальных полных подграфов и множества наибольших полных подграфов [8]. Результаты приведены в таблице, в которой использованы обозначения: $n$ – число вершин в графе, $m$ – число дуг, $s$ – средняя локальная степень вершин, $|N_{min}|$ – число вершин в множестве $N_{min}$, $Q$ – сумма весов всех вершин графа, $Q_{min}$ – сумма весов вершин множества $N_{min}$. Из таблицы хорошо видно, что эффективность рассмотренных алгоритмов повышается с уменьшением сложности графов, характеризуемой средней локальной степенью вершин.

Таблица

Результаты экспериментальных исследований

| Программа | $n$ | $m$ | $s$ | $\|N_{min}\|$ | $Q$ | $Q_{min}$ | $(Q - Q_{min})/Q$ |
|---|---|---|---|---|---|---|---|
| МАКСТРО (рис. 1 и 2) | 9 | 19 | 4,2 | 8 | 208 | 178 | 0,14 |
| МАКСПОДГРАФ (рис. 4) | 13 | 20 | 3,0 | 8 | 600 | 290 | 0,5 |
| МНАПОПОД | 24 | 33 | 1,37 | 14 | 356 | 106 | 0,7 |

Таким образом, предложенные алгоритмы позволяют существенно уменьшить затраты на построение моделей последовательных программ благодаря сокращению числа счетчиков на линейных участках и переходах между ними.

**Список литературы**

1. Губкин, А.Ф. Об определении временных характеристик последовательных программ / А.Ф. Губкин, В.А Матвеев // УСИМ. – 1991. – № 5. – С. 48–54.

2. Закревский, А.Д. К оптимизации динамического использования регистров в программах / А.Д. Закревский // Доклады АН БССР. – Т. 20, № 2. – 1976. – С. 127–129.

3. Поляков, А.С. О распределенной обработке программ в вычислительных сетях / А.С. Поляков // АВТ. – 1992. – № 2. – С. 45–49.

4. Поляков, А.С. Размещение фрагментов последовательных программ на процессорах распределенной системы / А.С. Поляков // Весці НАН Беларусі. Сер. фіз.-тэхн. навук. – 1999. – № 3. – С. 84–88.

5. Головкин, Б.А. Расчет характеристик и планирование параллельных вычислительных процессов / Б.А. Головкин. – М.: Радио и связь, 1983. – 272 с.

6. Joseph, D. Prefetching using markov predictors / D. Joseph, D. Grunwald // IEEE Transactions on Computers. – Vol. 48, № 2. – February 1999. – P. 121–133.

7. Chilimbi, T.M. Dynamic hot data stream prefetching for general-purpose programs / T.M. Chilimbi, M. Hirzel // Proc. of the ACM SIGPLAN 2002 Conference on Programming language design and implementation. – ASM Press, 2002. – P. 199–209.

8. Поляков, А.С. Нахождение максимальных полных подграфов / А.С. Поляков // Алгоритмы решения логико-комбинаторных задач: сб. науч. тр. – Минск: Ин-т техн. кибернетики АН БССР, 1975. – С. 32–42.

**Поступила 17.01.06**

*Объединенный институт проблем информатики НАН Беларуси,*
*Минск, Сурганова, 6*
*e-mail: paliichuk@bsu.by*

**A.S. Poljakov, V.E. Samsonov**

## COST MINIMIZATION FOR SEQUENTIAL PROGRAM MODEL CONSTRUCTION UNDER DISTRIBUTED REALIZATION

A cost minimization task has been considered for the process of determining the computation complexity of sequential programs. A weighted oriented graph is used as a program mathematical model. Its vertices correspond to linear program parts and arcs – to possible transactions between them, whereas weights of vertices and arcs – to the number of linear parts and transition executions occur during the program implementation. Algorithms for finding the minimal sets of graph vertices and arcs are proposed. Based on the given values the weights of all other vertices and arcs are computed.