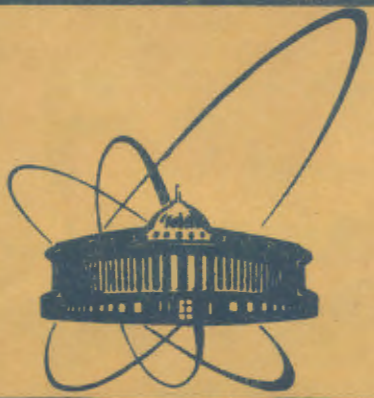

сообщения
объединенного
института
ядерных
исследований
дубна

Р11 - 13059

Е.Ю.Мазепа, Е.Д.Федюнькин

ОПТИМАЛЬНЫЕ СЕТИ С КВАЗИАДДИТИВНОЙ ФУНКЦИЕЙ КАЧЕСТВА

1980

1. ОСНОВНЫЕ ПРИНЦИПЫ

Рассмотрим связный симметрический граф $G=(X,U)$ с множеством вершин X и множеством ребер U /ребро задается парой своих граничных вершин/. С каждой вершиной $x_i \in X$ ассоциирован вес W_i, с каждым ребром $(x_i, x_j) \in U$ - вес W_{ij}. Вес представляет собой полную систему локальных характеристик вершины или ребра и может быть объектом любой природы - скаляром, вектором, битовым набором и т.п. Одну из вершин графа назовем корнем. Тогда любое дерево $T=(X,t)$, $t \subset U$ задает вполне определенный частичный порядок /1/ на множестве своих вершин /а следовательно - и на множестве ребер/, поскольку возникают понятия "ближе к корню", "дальше от корня". Сам корень выступает в качестве наименьшего элемента частично упорядоченного множества вершин. Иначе говоря, если указан корень, то дерево графа G можно рассматривать как ориентированное дерево с ребрами, направленными от корня. Легко заметить, что при этом каждой вершине /за исключением корня/ инцидентно в точности одно входящее ребро. Все ребра, инцидентные корню, - исходящие.

Определим "токи" j_{ki} на ребрах дерева и его качество F:

$$j_{ki} = W_i + \sum_{x_i < x_\ell} J(j_{i\ell}, W_{i\ell}) , \qquad /1/$$

$$F = \left| \sum_{(x_i, x_\ell) \in t} f(j_{i\ell}, W_{i\ell}) \right| . \qquad /2/$$

В /2/ суммирование производится по всем ребрам дерева, в рекуррентной формуле /1/ - по всем ребрам, исходящим из вершины x_i. J и f - заданные функции, не зависящие от выбора дерева. Что касается самой операции суммирования, это не обязательно арифметическое суммирование; допустима любая ассоциативная и коммутативная операция, типы операций в /1/ и /2/ могут быть различными. Очевидно, на ребрах, инцидентных листьям дерева, $j_{ki} = W_i$. Результат суммирования в /2/ определяется типом функции f и ее аргументов и, вообще говоря, может быть объектом произвольной природы, но F должна быть неотрицательным скаляром. Мы понимаем взятие модуля в /2/ как однозначную процедуру преобразования в неотрицательный скаляр/и не более/.

Равенства /1/ и /2/ вводят класс функций качества, который будем называть квазиаддитивным. Ограничения имеют не аналитический, а чисто конструктивный характер, задают способ построения F по заданным J и f. Вычисление квазиаддитивной функ-

ции качества сводится к элементарной трансформации /на одном ребре/ и может быть реализовано с помощью универсального алгоритма, не зависящего от конкретного вида функции.

Ограничения /1/, /2/ оставляют достаточно широкую свободу действий. Целый ряд практических задач попадает в наш класс. Один из важных примеров - электрическая сеть. Пусть W_i - заданные токи потребителей в вершинах графа, $W_{i\ell}$ - сопротивления ребер. Положим $J(j_{i\ell}, W_{i\ell}) \equiv j_{i\ell}$. Тогда /1/ определяет ток на ребре (x_k, x_i) дерева T. Полагая $f(j_{i\ell}, W_{i\ell}) \equiv j_{i\ell}^2 W_{i\ell}$, найдем, что /2/ определяет потери мощности на дереве T. Если заданы комплексные сопротивления ребер $W_{i\ell}^z$ и емкостные утечки $W_{i\ell}^c$, следует положить $J \equiv j_{i\ell} - iW_{i\ell}^c$, $f \equiv (j_{i\ell} - \frac{iW_{i\ell}^c}{2})^2 W_{i\ell}^z$ и понимать модуль в /2/ в обычном смысле. Если F - надежность дерева, а f - надежность одного ребра, сумму в /2/ следует понимать как произведение.

Задача оптимизации выглядит следующим образом: найти дерево T, для которого функция качества F удовлетворяет равенству

$$F = \operatorname{ext}_n F_n. \qquad /3/$$

Здесь $\operatorname{ext} - \min$ или $\max$, перебор осуществляется по всем деревьям исходного графа G.

Известно решение простейшей такой задачи /2/ - поиск дерева с экстремальной суммой весов ребер ($f(j_{i\ell}, W_{i\ell}) \equiv W_{i\ell}$). Задача решается последовательным исключением из исходного графа ребер с наихудшими весами / с одновременным контролем сохранения связности/. Если функция качества имеет более сложный вид, полный перебор деревьев для поиска абсолютного оптимума становится неизбежным. С этой целью мы воспользовались быстрым алгоритмом перебора всех частичных графов с заданным цикломатическим числом /3/. Если достаточно найти относительный оптимум, можно избежать полного перебора и решить задачу посредством релаксационного метода, который здесь не рассматривается. Однако разработанный нами общий подход полностью совместим с релаксационным методом.

2. СКАНИРОВАНИЕ ДЕРЕВА И ПРОБЛЕМНЫЕ ПРОЦЕДУРЫ

Естественный метод решения задачи /3/ - перебирать деревья, сравнивая их качества и запоминая лучшее из них. На первый взгляд, неясно, как найти качество конкретного дерева, поскольку "токи" $j_{i\ell}$, входящие в /2/, связаны рекуррентной формулой /1/. Ниже мы покажем, что эта формула легко разрешается, если сканировать дерево в направлении, обратном частичному порядку, т.е. от листьев в корню.

Пусть выбрано некоторое дерево $T=(X,t)$, $t \subset U$. Пусть L, M - массивы, ассоциированные с вершинами графа, причем $L(i)=W_i$, а элемент $M(i)$ равен степени вершины x_i в дереве T /т.е. равен числу ребер, инцидентных этой вершине/. Прибавим единицу к элементу $M(i)$, соответствующему корню дерева. Пусть ϕ - переменная, начальное значение которой пусто. Будем теперь сканировать список ребер дерева, обозначив через (x_i, x_ℓ) очередное ребро и придерживаясь следующих правил:

1. Если $M(i) \neq 1$ и $M(\ell) \neq 1$, то переходим к следующему ребру.
2. Если, например, $M(\ell) = 1$, выполняем операции:

$$M(\ell) := M(\ell) - 1, \quad M(i) := M(i) - 1, \qquad /4/$$

$$L(i) := L(i) + J(L(\ell), W_{i\ell}), \qquad /5/$$

$$\phi := \phi + f(L(\ell), W_{i\ell}). \qquad /6/$$

Ребро (x_i, x_ℓ) исключается из списка.

3. Если список ребер не пуст и мы достигли последнего ребра, начинаем сканирование сначала.

Сравнивая /5/ с /1/ и /6/ с /2/, найдем, что после окончания процедуры сканирования значение переменной ϕ равно качеству дерева /без "взятия модуля"/, а $L(i) = j_{ki}$, причем (x_k, x_i) - ребро, входящее в вершину x_i. Пару операций /5/, /6/, ассоциированную с одним ребром, назовем элементарной трансформацией.

При программной реализации задачи оптимизации удобно оформить элементарную трансформацию в виде отдельной процедуры. Необходима еще одна процедура - сравнение качеств двух деревьев, которая должна содержать в себе, в частности, операцию "взятия модуля". Две указанные процедуры /проблемные процедуры/ имеют простую логику и содержат в себе исчерпывающую информацию об устройстве конкретной функции качества. Остальное - перебор деревьев, формирование списка ребер, массивов M и L, определение порядка сканирования ребер, запоминание лучшего текущего дерева и т.п. - может быть оформлено в виде универсального алгоритма, не зависящего от конкретных свойств объекта оптимизации. Описанная ниже система IGRA реализует именно этот универсальный алгоритм. Написание проблемных процедур естественно отнести к компетенции конкретного пользователя системы. Длительная эксплуатация системы доказала жизнеспособность идеологии проблемных процедур. Она оказалась удачной еще и по следующим причинам. Представления пользователя о виде функции качества подвержены частым изменениям, и очень удобно, если обеспечен простой доступ к аппарату варьирования этой функции.

Идеология проблемных процедур может быть рекомендована для внедрения в самых различных областях программирования. Подобная идеология реализована, в частности, в программе FUMILI /4/ и в редакторе текстов для ЭВМ БЭСМ-6 /5/. Приведем еще один пример, показывающий силу этой идеологии. Любой алгоритм сортировки /например, сортировка Шелла/ сам по себе не связан с конкретными свойствами объектов сортировки. Объектами могут быть числа, битовые наборы, текстовые строки, файлы на диске и т.п. Сортировка может осуществляться по алфавитному признаку, по величине числа, по способу расположения битов в слове и т.п. Тем не менее можно создать одну единственную программу сортировки на все случаи жизни, если ввести понятие проблемной процедуры, которая по номерам двух объектов производит их сравнение и/если нужно/ перестановку.

3. ОБЩАЯ ОРГАНИЗАЦИЯ СИСТЕМЫ IGRA

Система реализована на ЭВМ БЭСМ-6, выполнена в виде единого программного блока с десятью входами и занимает 940 слов в оперативной памяти. Информация об исходном графе упакована по одному ребру в слово. В то же слово упакованы динамические и статические признаки, относящиеся к данному ребру, и, в частности, элемент флагового массива, на котором реализован алгоритмический стек процедуры перебора /3/. Система написана на языке MADLEN. Ниже перечислены имена и параметры входов.

FUNCTION IGRA (WS)

FUNCTION IOPT (WS, V, R, USER1, USER2, RET)

SUBROUTINE OPENG (WS, CY)

SUBROUTINE STARTG (WS, CY)

FUNCTION IP (WS, X1, X2, FIX, START)

SUBROUTINE IGRAS (WS, I)

FUNCTION IUP (WS, I, REG)
(IW, X1, X2)

SUBROUTINE INFOPT (WS, ϕ1, ϕ2, I)

FUNCTION IFRAG (WS)
(IW)

SUBROUTINE ERRG (WS)

Пусть NV - число вершин, NR - число ребер исходного графа. Вершины графа нумеруются целыми числами, максимальное из которых не должно превосходить (NR+1). WS - рабочий массив размерности (NR+3) для целей процедуры IGRA или размерности (NR+2NV+30) для целей процедуры IOPT. Ребро /пара вершин/ упаковано вместе с атрибутами в одном слове массива WS. Список ребер начинается с WS(3) и оканчивается словом 0B.

Упаковку производит процедура IP. CY - целое, цикломатическое число или номер вершины-корня. V - массив размерности NV, ассоциированный с вершинами графа. В массиве V содержатся веса W_i. R - массив размерности NR, ассоциированный с ребрами графа. В массиве R содержатся веса W_{il} /или их адреса/. Порядок расположения информации в массиве R должен соответствовать порядку расположения ребер в массиве WS. USER1 - имя функции /типа INTEGER или REAL /, реализующей проблемную процедуру элементарной трансформации, к которой IOPT обращается с параметрами USER1(R, L2, L1, ϕ). Здесь R - элемент из массива R; L1, L2 - элементы из массива L /см. раздел 2/; ϕ - массив из двух слов, накопитель качества, первоначально содержит 0B. Направление от L1 к L2 соответствует направлению от корня дерева. Функция USER1 может модифицировать L1 и ϕ. Для каждого дерева массив L формируется заново процедурой IOPT, он является частью массива WS. Если обнаружено, что текущее дерево абсолютно плохое, функция USER1 должна возвращать значение нуль. USER2 - имя функции /типа INTEGER или REAL /, реализующей проблемную процедуру сравнения качеств двух деревьев, к которой IOPT обращается с параметрами USER2(ϕ_{new}, ϕ_{old}). Функция должна возвращать значение нуль, если ϕ_{new} хуже, чем ϕ_{old}.

RET - целое, определяет момент возвращения из IOPT в вызывающую программу. X1, X2 - целые переменные, в которых содержатся /или в которые пересылаются/ номера пары вершин, составляющих ребро. FIX - целый нуль или единица, атрибут фиксации. Если FIX=1, данное ребро не может быть исключено. START - целый нуль или единица, атрибут эталона. Если START=1, данное ребро исключено в эталонном частичном графе. Процесс перебора перестраивается таким образом, что эталон оказывается первым найденным ν-графом. Здесь ν - цикломатическое число искомого частичного графа. Если для всех ребер исходного графа START=0, процедура IGRA выбирает в качестве эталонного любой ν-граф. I - целое число, ассоциированное с одним из графов, хранящихся в системе. I может принимать значения 1-10 для процедуры IGRAS, (-1)-13 для процедуры IUP, 0-10 для процедуры INFOPT. REG - целый нуль или единица, определяет способ выдачи информации. Если REG=0,

выдаются ребра частичного графа. Если REG=1, выдаются исключенные ребра исходного графа. IW - информационное слово, формируемое функциями IUP и IFRAG. $\phi 1$, $\phi 2$ - первое и второе слово накопителя качества ϕ, засылаются процедурой INFOPT.

4. ЗАСЫЛКА ИНФОРМАЦИИ В СИСТЕМУ

Любая деятельность начинается с обращения к процедуре OPENG с цикломатическим числом /если в дальнейшем будет использоваться IGRA / или номером вершины-корня /если в дальнейшем будет использоваться IOPT /. OPENG полностью уничтожает информацию в массиве WS путем модификации первых трех его слов: WS(1):=WS(3):=0B; WS(2):=CY. После этого обращаемся в цикле к процедуре IP столько раз, сколько ребер в исходном графе. При каждом обращении IP упаковывает в массив WS одно ребро.

Пустое слово WS(1) служит для процедур IGRA, IOPT сигналом о необходимости начать процесс перебора сначала. В любой момент можно вернуть процесс перебора к исходной точке /с засылкой нового CY /, обратившись к процедуре STARTG, которая, в отличие от OPENG, не уничтожает списка ребер в массиве WS. После обращения к STARTG список ребер можно расширить, обращаясь нужное число раз к IP.

Функция IP располагает ребра в массиве WS в том порядке, в котором они приходят. Значение функции - внутренний номер ребра в массиве WS. Например, IP=1 означает, что данное ребро заслано в слово WS(3). Этим обстоятельством можно воспользоваться, чтобы согласованно рассылать информацию в массив R. Пусть, например, X1, X2 содержат номера граничных вершин ребра, XX - соответствующий этому ребру вес. Рассылка /на фортране/ может выглядеть так:

```
IND = IP(WS, X1, X2, FIX, START)
R(IND) = XX.
```

5. ПРОЦЕДУРЫ IGRA, IOPT, IGRAS

Функция IGRA является программной реализацией алгоритма перебора всех частичных графов [3] с некоторыми модификациями. Во-первых, появляется возможность фиксировать ряд ребер исходного графа, запретив их исключение с помощью атрибута FIX; во-вторых, появляется возможность задать эталонный ν-граф с помощью атрибута START. Последнее особенно полезно, если число частичных графов велико, так что мы не можем их перебирать и поэтому хотим исследовать частичные графы "в окрестности" эталона. Заметим, что использование атрибута FIX позволяет, в частности, сократить число результирующих ν-графов. В случае больших графов полезно организовать интерактивную работу с программой, чтобы человек мог использовать свои интуитивные возможности, манипулируя атрибутами.

Функция IOPT является программной реализацией алгоритма оптимизации, изложенного в разделах 1 и 2 настоящей работы. IOPT содержит в себе обращение к функции IGRA. Массивы R и V должны быть сформированы пользователем до первого обращения к IOPT. Функции USER1 и USER2 также должны быть созданы пользователем. Информация в массивах R и V может быть любая, поскольку единственная процедура, использующая эту информацию, - USER1.

При первом обращении к IGRA или IOPT производится диагностика ошибок. Признак ошибки - нулевое значение соответствующей функции. В этом случае в слове WS(2) появляется номер ошибки. Подробный текст диагностики будет распечатан, если обратиться к процедуре ERRG. При первом обращении к IOPT производится перестановка ребер исходного графа в соответствии с их распределением по квазициклам. Процедура IOPT обеспечивает при этом согласованную перестановку элементов массива R. Перед первым обращением к IGRA процедура IOPT засылает в нуль слово WS(2), чтобы обеспечить генерацию всех деревьев /0-графов/ исходного графа.

Возвращение в вызывающую программу из функции IGRA происходит после того, как сформирован очередной ν-граф. Значением функции является порядковый номер этого ν-графа. Вся информация о состоянии процесса перебора хранится в массиве WS. Следовательно, для полного перебора нужно обратиться к функции IGRA столько раз, сколько существует различных ν-графов. Нулевое значение функции IGRA /если это не первое обращение/ сигнализирует о том, что последний ν-граф был выдан в предыдущем обращении. Возвращение в вызывающую программу из функции IOPT происходит после того, как обработано RET деревьев. Если RET=0, возвращение происходит после того, как произошло очередное улучшение /т.е. найдено дерево лучшее, чем все деревья, исследованные до текущего момента/. Значение функции IOPT - порядковый номер последнего улучшения. Нулевое значение функции IOPT /если это не первое обращение/ сигнализирует о том, что все деревья исчерпаны, и, следовательно, процесс оптимизации завершен.

В массиве WS имеется генеральный стек, в котором можно поместить десять ν-графов. Чтобы запомнить последний сгенерированный ν-граф, на уровне I /здесь I может принимать значения 1-10/ нужно обратиться к процедуре IGRAS:

```
CALL IGRAS (WS, I).
```

При этом ν-графы, которые находились на уровнях (I,9), смещаются на уровни (I+1,10), а информация с десятого уровня теряется.

При работе с IOPT обращаться к IGRAS не следует, IOPT делает это сама. После окончания процесса оптимизации десять лучших деревьев находятся в генеральном стеке, располагаясь по уровням в порядке ухудшения их качеств.

После первого обращения к IGRA или IOPT и в дальнейшем в слове WS(2) находится номер последнего сгенерированного ν-графа, поэтому после окончания процесса перебора в WS(2) оказывается число ν-графов исходного графа.

6. ВЫВОД ИНФОРМАЦИИ ИЗ СИСТЕМЫ

Вывод информации осуществляется с помощью функции IUP. Инициация вывода

IW = IUP(WS, I, REG).

После инициации обращаемся в цикле к IUP со следующими параметрами:

IND = IUP(IW, X1, X2).

Каждый раз в X1 и X2 выдаются граничные вершины очередного ребра. Значение функции - порядковый номер ребра в массиве WS, поэтому R(IND) является элементом массива R, соответствующим данному ребру. Нулевое значение IUP /при этом в X1 и X2 также засылаются нули/ сигнализирует о том, что последнее ребро получено в предыдущем обращении. Ребра выдаются в порядке возрастания номеров их вершин. Функцию IUP можно использовать в любой момент в любом месте программы и, в частности, в проблемной процедуре.

Может быть выдан последний найденный ν-граф (I=0) или содержимое одного из десяти уровней генерального стека (I=1-10), или эталон (I=12). В указанных случаях, если REG=0, то выдаются ребра ν-графа; если REG=1, то выдаются исключенные ребра /т.е. ребра исходного графа, отсутствующие в данном ν-графе/. Могут быть выданы фиксированные (I=11,REG=1) и нефиксированные (I=11, REG=0) ребра. Может быть выдан исходный граф /I=-1, REG игнорируется/.

При оптимизации /т.е. когда мы работаем с IOPT / можно получить качества деревьев, соответствующих I=0-10, если вызвать процедуру INFOPT. Ее можно вызвать из любого места программы и, в частности, из проблемной процедуры.

7. ДИАГНОСТИКА ОШИБОК

Ниже перечислены номера ошибок и диагностические тексты, которые распечатывает процедура ERRG:

0. ПУСТОЙ ГРАФ.
0. НЕДОПУСТИМАЯ ФИКСАЦИЯ.
1. ОБНАРУЖЕНЫ КРАТНЫЕ РЕБРА.
2. ОБНАРУЖЕНЫ ПЕТЛИ.
3. НЕДОПУСТИМАЯ ВЕРШИНА.
4. ОБНАРУЖЕНО ВИСЯЧЕЕ РЕБРО.
5. ОМЕРЗИТЕЛЬНЫЙ ГРАФ - НЕСВЯЗНЫЙ ГРАФ.
6. НЕДОПУСТИМОЕ ЦИКЛОМАТИЧЕСКОЕ ЧИСЛО.
7. НЕКОРРЕКТНЫЙ ЭТАЛОН.
8. НЕДОПУСТИМАЯ ОПТИМИЗАЦИЯ.
9. НЕДОПУСТИМЫЙ ИСТОЧНИК.

Диагностика "НЕДОПУСТИМАЯ ФИКСАЦИЯ" появляется, если невозможно получить ни одного искомого частичного графа без исключения фиксированного ребра. Петля - ребро типа (x_i, x_i). Ошибка 3 возникает, если обнаружена вершина, номер которой превышает число ребер исходного графа плюс единица. Ошибка 4 появляется, если одна из вершин имеет номер нуль. Ошибка 6 возникает, если не существует частичного графа с заданным цикломатическим числом. Ошибка 7 появляется, если эталон - неправильный граф, т.е. имеет не то цикломатическое число или несвязный, или не содержит в себе всех фиксированных ребер. Ошибка 8 возникает, если в процессе оптимизации значение функции USER1 всегда было нулевым, т.е. все деревья объявлены абсолютно плохими. Ошибка 9 появляется, если номер корня не соответствует ни одной вершине графа /источник - то же самое, что корень/.

Ошибка 5 фиксируется следующим образом. Выстраивается некоторый максимальный связный фрагмент / IOPT начинает построение фрагмента от корня/. Если найдется хотя бы одна вершина, не принадлежащая этому фрагменту /т.е. если фрагмент не совпадает с исходным графом/, то фиксируется ошибка - мы имеем граф, который принято называть омерзительным. Ребра, принадлежащие связному фрагменту (REG=1), или ребра, не принадлежащие связному фрагменту (REG=0), можно получить, обратившись к IUP с I=13. Вершины, не принадлежащие связному фрагменту, можно получить с помощью функции IFRAG. Инициация вывода

IW = IFRAG(WS).

После инициации обращаемся в цикле к IFRAG(IW). Значением функции является номер очередной вершины или нуль, если последняя вершина была выдана в предыдущем обращении. Напомним, что здесь обсуждается связность и несвязность исходного графа.

При диагностике ошибок всякая деятельность прекращается, как только найдена первая ошибка, другие ошибки не ищутся.

Авторы признательны С.С.Лебедеву, А.П.Сапожникову, И.Н.Силину, которые принимали участие в обсуждении настоящей работы.

ЛИТЕРАТУРА

1. Birkhoff G. Lattice Theory. New York, 1948.
   Биркгоф Г. Теория структур. ИЛ, М., 1952.
2. Kruskal J.B. (Jr). Proc.Am.Math.Soc., 1956, 7, p.48.
3. Мазепа Е.Ю., Силин И.Н., Федюнькин Е.Д. ОИЯИ, Р5-12874, Дубна, 1979.
4. Силин И.Н. Программа сравнения гипотез по r и r^2-критериям. KFKI-74-34, Budapest, 1974, p.181-189.
5. Волков А.И. Редактор текстов. ИАЭ-2351, М., 1974.

Рукопись поступила в издательский отдел
29 декабря 1979 года.

Д6-11574	Сборник аннотаций XV совещания по ядерной спектроскопии и теории ядра. Дубна, 1978.	2 р. 50 к.
Д3-11787	Труды III Международной школы по нейтронной физике. Алушта, 1978.	3 р. 00 к.
Д13-11807	Труды III Международного совещания по пропорциональным и дрейфовым камерам. Дубна, 1978.	6 р. 00 к.
	Труды VI Всесоюзного совещания по ускорителям заряженных частиц. Дубна 1978. /2 тома/	7 р. 48 к.
Д1,2-12036	Труды V Международного семинара по проблемам физики высоких энергий. Дубна 1978.	5 р. 00 к.
Р18-12147	Труды III совещания по использованию ядерно-физических методов для решения научно-технических и народнохозяйственных задач.	2 р. 20 к.
Д1,2-12450	Труды XII Международной школы молодых ученых по физике высоких энергий. Приморско, НРБ, 1978.	3 р. 00 к.
Р2-12462	Труды V Международного совещания по нелокальным теориям поля. Алушта, 1979.	2 р. 25 к.
Д2-11707	Труды XI Международной школы молодых ученых по физике высоких энергий и релятивистской ядерной физике. Гомель, 1977.	6 р. 00 к.

Заказы на упомянутые книги могут быть направлены по адресу:

101000 Москва, Главпочтамт, п/я 79,

издательский отдел Объединенного института ядерных исследований