ПРИЛОЖЕНИЕ 3

БИБЛИОТЕКА ЭНДШПИЛЯ ПРОГРАММЫ «ПИОНЕР» (ИСПОЛЬЗОВАНИЕ ОПЫТА ПРОШЛОГО ПО СПРАВОЧНОМУ МЕТОДУ И МЕТОДУ СТРЕМЛЕНИЯ)

А. Д. Юдин

3.1. Введение. Как отмечалось выше, при составлении программы «Пионер» была поставлена задача сделать ее содержание по возможности близким к содержанию «программы» шахматного мастера. Решению этой задачи служит создание информационной системы «Опыт прошлого».

Важнейшим фактором при решении вопроса о целесообразности создания любой информационно-справочной системы является эффективность использования этой системы. Только в случае, если есть основания к ее успешному использованию, если экспериментально доказано, что вновь созданная система приведет к эффективному решению задач, стоящих перед исследователями, тогда только и необходимо создавать такую информационно-справочную систему.

В нашем случае речь идет о целесообразности создания информационно-справочной системы «Опыт прошлого» шахматной программы для ЭВМ при моделировании мышления шахматиста. Мастер, играя в шахматы, систематически обращается к опыту прошлого в любой стадии партии. Он делает это в тех узлах дерева перебора, где это целесообразно. Отсюда следует, что для повышения эффективности использования опыта прошлого, во-первых, число узлов дерева перебора должно быть невелико и, во-вторых, программа использования опыта прошлого должна быть организована так, чтобы это требовало малого расхода ресурсов ЭВМ. Если удовлетворение второму требованию целиком и полностью находится в руках создателей информационной системы «Опыт прошлого», то первое требование неразрывно связано с принятым алгоритмом шахматной игры. И при создании шахматной программы по алгоритму М. Ботвинника мы имеем дело с небольшим «человеческим» деревом перебора.

Итак, «программу» шахматиста можно условно разделить на две части: 1) программу поиска хода в оригинальной ситуации; 2) библиотеки дебюта, миттельшпиля и эндшпиля и программы пользования этими библиотеками.

Разделение условно потому, что иногда эти две части работают параллельно, в сочетании друг с другом. При переборе в программе поиска хода часто используются библиотечные сведения.

В данном приложении будут рассмотрены вопросы, связанные с созданием библиотеки эндшпиля шахматной программы «Пионер» и алгоритмов пользования этой библиотекой по справочному методу и методу стремления, а также программная реализация указанных алгоритмов.

Поскольку в программе «Пионер» фигуры передвигаются по траекториям в соответствии с принятой целью игры, то задача пользования библиотекой эндшпиля может быть решена по аналогии с действиями шахматного мастера. Мастер, когда он играет в шахматы, не только ищет совпадение позиций из партии (или из дерева перебора) с позициями из библиотеки, но и стремится получить эти библиотечные позиции (стремление к библиотечной позиции будет рассмотрено в п. 3.10—3.17). Так будет действовать и «Пионер»; после нахождения в библиотеке похожей (близкой) и выгодной позиции, он будет искать траектории для фигур и передвигать фигуры так, чтобы эту выгодную позицию получить. Как только такая позиция будет достигнута, произойдет совпадение позиции из дерева перебора с позицией из библиотеки, оценка позиции станет известной и вариант оборвется.

3.2. Постановка задачи. Обычно в справочниках по эндшпилю приводятся позиции и соответствующие им варианты. Но сильный мастер, когда он играет эндшпиль, как правило, не помнит эти варианты. Он помнит опорные, узловые позиции, их оценки и, если это необходимо, трудные первые ходы. Все остальное мастер находит при помощи своего алгоритма поиска хода.

Поэтому представляется целесообразным включать в библиотеку не варианты, а узловые позиции, их оценки и первые ходы (если ходы трудные). Это существенно упрощает методику составления библиотеки и уменьшает объем запасенной информации [4].

Таким образом, перед мастером (или программой) стоит следующая задача. Имеется конкретная эндшпильная позиция, возникшая в партии или в каком-либо варианте перебора. Необходимо при помощи библиотеки оценить (выиграно, ничья, проиграно) данную позицию при очереди хода за той или иной стороной и указать наилучший первый ход (если это необходимо).

3.3. Конфигурации. *Конфигурацией* позиции из N фигур будем называть набор разностей координат $\Delta_1, \Delta_2, \ldots, \Delta_{N-1}$, где $\Delta_i = L_1 - L_{i+1}$ для $i = 1, 2, \ldots, N-1$; L_i — линейная координата i-й фигуры позиции (может принимать значения от 1 до 64).

Возьмем какую-нибудь позицию технического эндшпиля. Далее, зафиксировав относительное расположение фигур (конфигурацию), будем сдвигать позицию относительно вертикальной и горизонтальной осей координат всеми возможными способами, не выходя при этом за пределы доски, а также не нарушая шахматных правил с точки зрения возможного расположения пешек. Получаем множество позиций данной конфигурации. При составлении программы задача состоит в том, чтобы записать данное множество

позиций (их бывает до 40 в множестве) вместе с их решением (оценки, первые ходы) в библиотеку эндшпиля в удобном для работы программы виде.

3.4. Явление «краевого эффекта». Формулы разбиения. Возможность компактной записи требуемой информации возникает в связи с явлением, которое получило название «краевой эффект». Было установлено, что указанное множество позиций нетрудно разбить на несколько непересекающихся (с точки зрения оценок и конфигураций первых ходов, решающих позиции) подмножеств:

— позиции с влиянием одного из вертикальных краев доски;

— позиции с влиянием одного из горизонтальных краев доски;

— позиции с совместным влиянием горизонтального и вертикального краев доски (так называемые «угловые позиции»);

— все остальные позиции множества, когда отсутствует влияние краевого эффекта.

Оценки и первые ходы (для данной конфигурации) внутри каждого подмножества позиций остаются неизменными. Более того, поскольку все множество характеризуется неизменной конфигурацией, т. е. взаимным расположением фигур, а функция оценки имеет лишь три значения (выиграно, ничья, проиграно), то для большинства конфигураций некоторые из вышеуказанных подмножеств можно объединить из-за совпадения оценок и первых ходов. Часто также встречаются пустые подмножества, не содержащие ни одной позиции.

Таким образом, чтобы распознать любую позицию множества, достаточно иметь в библиотеке лишь одну позицию, например из углового подмножества (часто, но далеко не всегда, оно «вырождается» в одну позицию), а также формулы разбиения множества, которые характеризуют границы изменения оценок. Подробное решение задачи для одного множества будет показано ниже на примере.

3.5. Симметрии. Большое значение в программе имеет также использование симметрий:

— *фланговой симметрии* — отражения позиции относительно вертикальной оси доски;

— *симметрии цвета* — отражения позиции относительно горизонтальной оси доски, сопровождающегося переменой цвета фигур;

— *диагональных симметрий* — отражений позиции относительно диагоналей a1—h8 и h1—a8 (действуют в позициях без пешек).

Каждое поле доски характеризуется своими двумерными координатами x, y или линейной координатой L, которые связаны между собой:

$$L=8(y-1)+x.$$

Например, поле f5 имеет координаты: $x=6$, $y=5$ или $L=38$.

Опуская несложные преобразования, укажем формулы для

симметрирования. Пусть x, y, L — координаты поля, на котором стояла фигура до симметрирования. Тогда для симметрии флангов

$$L_{\text{фл}}=L+9-2x;$$

для симметрии цвета

$$L_{\text{цв}}=L+8(9-2y);$$

для диагональных симметрий

$$L_{\text{a1–h8}}=8L-63y+56 \text{ и } L_{\text{h1–a8}}=63y+9-8L.$$

Укажем, что для любого поля доски

$$L_{\text{a1–h8}}+L_{\text{h1–a8}}=65.$$

Это и некоторые другие полученные соотношения используются для образования симметрированных конфигураций.

Указанные симметрии используются как отдельно, так и совместно, в соответствии с условиями их применения. Так, позиция Белые: Крe3, Лh1: Черные: Крg5, Сh4, помещенная в библиотеку прямым или косвенным (с помощью формул, учитывающих краевой эффект) образом, характеризует еще 15 позиций, связанных с вышеприведенной совместным использованием всех трех симметрий.

3.6. Структура библиотеки. Классы. Кодирование информации. Структура библиотеки выглядит следующим образом. Все позиции технического эндшпиля разбиты по материалу на 31 класс. Например, «Король с пешкой против короля»; «Король, конь и пешка против короля и слона» и т. п.

Таким образом, внутри каждого класса соотношение материала в позициях неизменно.

Класс, помещенный в библиотеку, представляет собой прямоугольную матрицу размером $m\times n$, где m — количество позиций в классе; $n=\text{ENT}[(N+1)/2]+2$; N — количество фигур в каждой позиции данного класса.

Каждая строка матрицы соответствует одной позиции. Покажем кодирование информации на примере позиции

Белые: Крf2, п.g2; Черные: Крc6, п.h4.

Решение позиции выглядит следующим образом. При ходе белых выигрывает 1. Крf2—g1, при ходе черных — ничья: 1 ... ...h4—h3. Пусть эта позиция имеет номер i в классе «Король с пешкой против короля с пешкой».

Оценка r может принимать следующие значения:

$$r=\begin{cases}0 \text{ — выиграно,}\\ 1 \text{ — ничья,}\\ 2 \text{ — проиграно.}\end{cases}$$

Первые $n-2$ позиции (имеются в виду места в строке матрицы, а не шахматные позиции) в строке образуют попарно объединен-

ные координаты фигур. В двух последних столбцах стоят пятизначные (в общем случае) числа — ходы с приписанными впереди оценками. Таким образом, элементы i-й строки нашей матрицы соответственно равны: M_{i1}=1415; M_{i2}=4332; M_{i3}=1407; M_{i4}==13224.

В классах с нечетным N в $(n-2)$-м столбце стоят двузначные координаты N-й фигуры позиций.

Каждой позиции класса соответствуют формулы разбиения множества, символом конфигурации которого является данная позиция, а также формулы коррекций оценок и первых ходов (см. пример ниже).

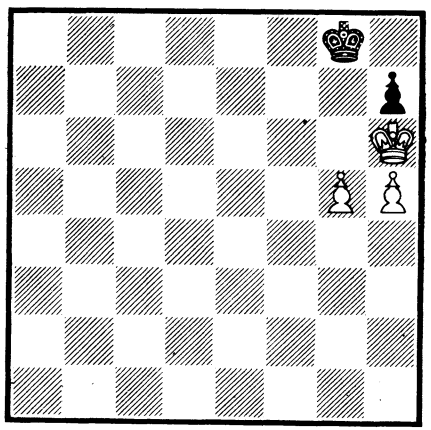

Рис. 37. Пример позиции

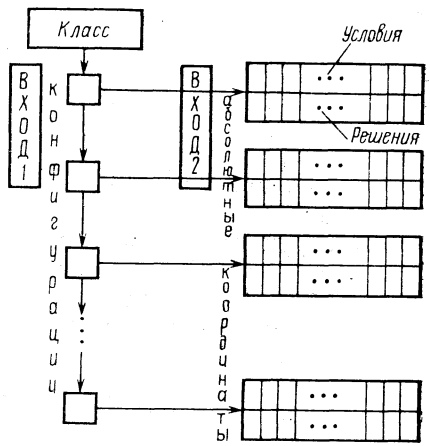

Рис. 38. Схема поиска информации в двумерной таблице с субординацией входов

При указанной структуре пополнение библиотеки новыми позициями (или исключение позиций из библиотеки) означает включение (исключение) строки или строк в (из) соответствующую матрицу-класс.

Рассмотрим поиск по справочному методу и действие краевого эффекта на примере простой конфигурации (см. позицию на рис. 37 и таблицу). Позиция и таблица, кроме последней колонки (приведенной для наглядности), «записаны» в библиотеке и соответствуют позиции на диаграмме в классе «Король и две пешки против короля и пешки».

Использованием симметрии флангов и цвета каждая позиция множества в процессе поиска преобразуется в четыре позиции.

3.7. Организация информации в виде двумерных таблиц с субординацией входов. Явление краевого эффекта позволяет расположить всю справочную информацию об эндшпиле в некоторое количество двумерных таблиц с субординацией входов.

Двумерной таблицей с субординацией входов назовем таблицу с двумя входами, один из которых является независимым, а другой зависит от первого.

Таблица

Характеристики множества позиций (к диаграмме позиции на рис. 37)

Номер подмножества	Способ получения подмножества из заложенной позиции	Оценка позиции		Первый ход		Количество позиций в подмножестве
		при ходе белых	при ходе черных	белых	черных	
1	Сдвиг вниз на 1—3 поля	ничья		$\backsim$[1]	$\backsim$	3
2	Подмножества нет[2]					0
3	Совпадает с заложенной позицией	ничья	выиграно у белых	$\backsim$	$\backsim$	1
4	Сдвиг влево на 1—6 полей и вниз на 1—3 поля	Выиграно у белых		$l_0^{БК} l_1^{БК}$, где[3] $l_1^{БК} = l_0^{БК} + 9$	$\backsim$	18

[1] $\backsim$ означает, что указание первого хода необязательно; [2] 2-е подмножество для данного примера „пустое", т. е. не содержит ни одной позиции; [3] $l_0^{БК}$ — начальное положение (координата) белого короля, $l_1^{БК}$ — положение белого короля после хода.

В нашем случае каждому классу с характерным соотношением материала соответствует одна двумерная таблица. На рис. 38 представлена структурная схема программы поиска информации в такой таблице. Первым, независимым (или безусловным), входом является поиск по конфигурации, т. е. поиск совпадения относительных координат позиции из партии (перебора) с относительными координатами одной из библиотечных позиций-символов конфигураций. В том и только в том случае, если совпадение обнаружено, управление передается подпрограмме поиска по второму входу таблицы, т. е. будет найдено решение уже в абсолютных координатах. Следует отметить, что эта часть таблицы (бо́льшая ее часть) существует не в явном виде, а лишь в виде соответствующих формул разбиения и воспроизводится по этим формулам тогда и только тогда, когда найдено совпадение относительных координат [6].

3.8. Алгоритм пользования библиотекой эндшпиля (поиск точного совпадения). Комплекс подпрограмм, реализующий алгоритм

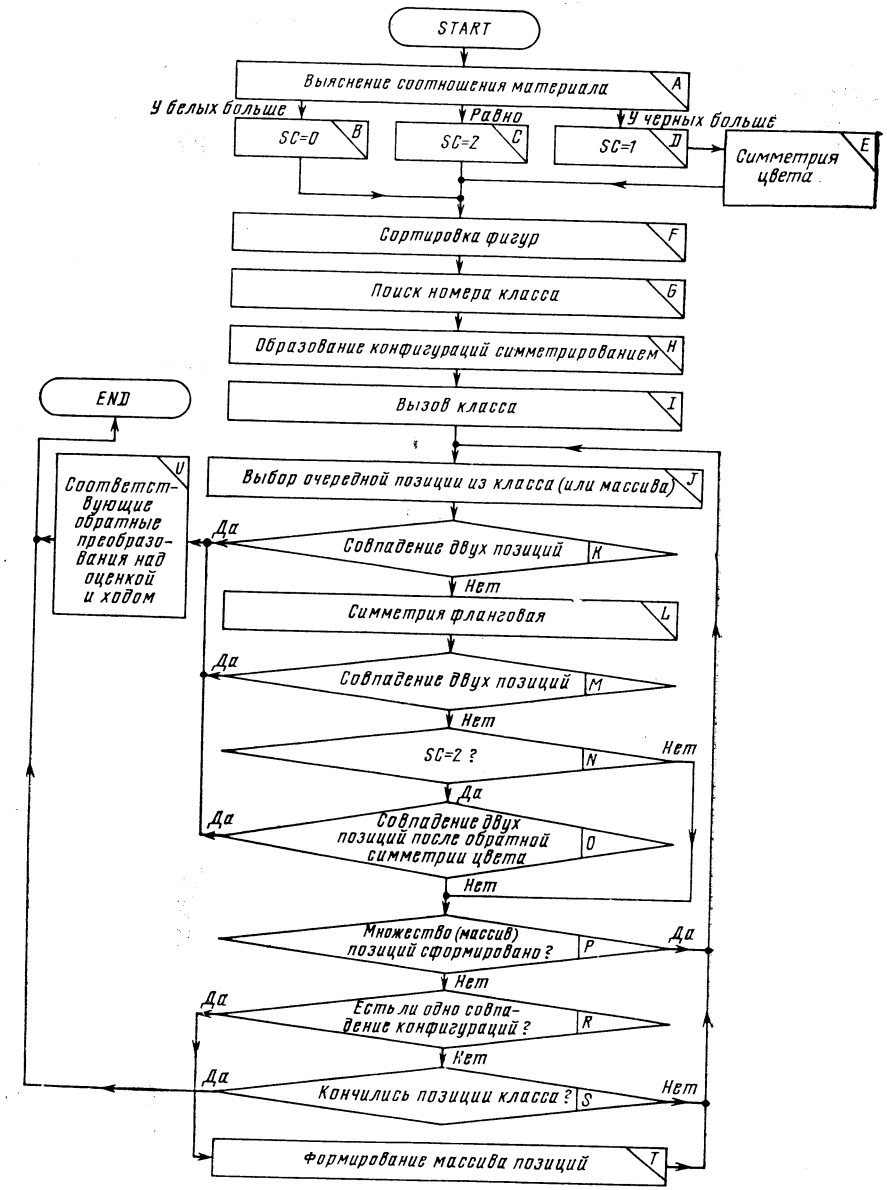

Рис. 39. Блок-схема алгоритма пользования библиотекой эндшпиля (поиск точного совпадения)

пользования, включается тогда, когда в позиции партии или какого-либо варианта перебора впервые встретилось соотношение материала, отвечающее (с точностью до перемены цвета фигур) соотношению материала одного из классов библиотеки.

Рассмотрим блок-схему алгоритма пользования библиотекой эндшпиля (рис. 39). (На схеме для упрощения не указаны процедуры диагональных симметрий.)

Процедура A выясняет соотношение материала сторон, что необходимо для решения вопроса о возможности симметрии цвета. Все дальнейшие процедуры предусматривают материальный перевес белых или равенство; так построена и сама библиотека. Поэтому, если в позиции партии или перебора у черных материальный перевес, то производится симметрия цвета (процедура E) и находится решение для позиции с переменой цвета фигур, а после нахождения решения процедура U производит обратное преобразование, на необходимость которого укажет присвоенное значение $SC=1$ (процедура D). Если на доске или в позиции перебора материальное равенство, то можно искать решение как для исходной, так и для симметрированной по цвету позиции, о чем говорит значение $SC=2$ (процедура C).

Процедура F приводит исходную позицию к виду, удобному для сравнения с библиотечными, т. е. строками матрицы-класса. Здесь происходит сортировка фигур и соответствующее кодирование исходной позиции. После этого вычисляется номер класса (процедура G), который процедура I вызовет в оперативную память.

Рассмотрим процедуру G. Для всех конфигураций данного класса существует некоторая общая характеристика, названная шаблоном, определяющаяся конкретным соотношением фигур.

Шаблоном позиции A назовем массив U из 12 элементов, где $U(1)=1$; $U(2)$ — количество белых ферзей в позиции A; ... ... $U(12)$ — количество черных пешек в позиции A.

Первые шесть элементов этого массива — $U_w(6)$ — назовем белым шаблоном; последние шесть — $U_b(6)$ — черным шаблоном. Такие шаблоны запасены для каждого класса и являются его «материальной» характеристикой.

Далее для позиции партии (перебора) «изготовляется» такой же шаблон. Поиск номера класса сравнением шаблонов показан на рис. 40. Этот поиск осуществляется с учетом симметрии цвета, на возможность или необходимость которой указывает значение SC.

Совпадение шаблона исходной позиции с одним из шаблонов классов автоматически указывает номер класса, который процедура I вызовет в оперативную память машины.

Процедура H из исходной позиции симметрированием образует всевозможные конфигурации.

Когда в первый раз управление передается процедуре J, то процедуры K, L, M, N и O производят поиск точного, с учетом симметрирования, совпадения исходной позиции с очередной позицией-символом множества в классе (поиск по первому входу в соответствующей двумерной таблице); при успешном завершении поиска управление передается процедуре U, которая производит обратные (с учетом сделанных симметрий) преобразования над найденными оценкой и ходом.

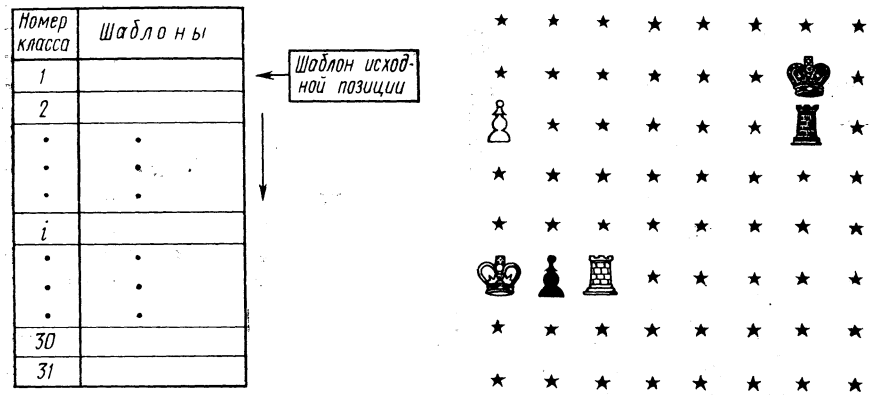

Рис. 40. Схема поиска **номера класса** сравнением шаблонов


```
                                MASSIV= 32
 1  A2  =   B9-B9   =   B9-B9
 2  B2  =   B9-B9   =   B9-B9
 3  A3  +-  C3-G3   =   G6-A6
 4  B3  +-  D3-H3   =   H6-B6
 5  A4  +-  C4-G4   =   G7-A7
 6  B4  +-  D4-H4   =   H7-B7
```

Рис. 41. Пример массива позиций, заложенных вместе с их решением в библиотеку

Предположим, а это наиболее вероятный случай, что точного совпадения (с учетом симметрий) не обнаружено. Тогда проверка P, которой управление передается впервые, дает отрицательный ответ и передает управление проверке R; последняя решает вопрос о необходимости восстановления или образования множества по позиции-символу, т. е. по той позиции из библиотеки, с которой мы сейчас имеем дело.

Для того чтобы решение существовало в пределах множества для данной позиции класса, необходимо и достаточно, чтобы конфигурация этой позиции совпала хотя бы с одной из симметрированных конфигураций позиции партии или перебора. Доказательство как необходимости, так и достаточности этого утверждения элементарно вытекает из определения конфигурации, правил симметрирования и способа образования множества позиций данной конфигурации.

Таким образом, при положительном результате проверки R мы имеем полную гарантию того, что массив или множество позиций зря сформированы не будут (процедура T). Это означает, что после проверки всех (в общем случае) позиций множества, сформированного процедурой T, управление неизбежно будет передано процедуре U вследствие положительного результата одной из проверок K, M или O. Это и есть поиск по второму входу в соответствующей двумерной таблице

Следует отметить, что при помощи формул разбиения и коррекции оценок и ходов процедура T образует множество позиций уже с их решениями, которые записываются в двух последних столбцах матрицы. После необходимых обратных преобразований над оценкой и ходом (процедура U) задача решена.

Предположим теперь, что проверка R дает отрицательный результат. Тогда управление передается процедуре J, выбирается очередная позиция-символ и т. д.

Если мы перебрали все позиции-символы в классе и ни разу проверка R не дала положительного результата, то по окончании всех позиций-символов в классе проверка S дает положительный ответ, а это означает, что точного решения не существует, т. е. поиск точного совпадения по конфигурации класса успешно не завершился и, следовательно, искомая позиция в библиотеке не найдена. Это происходит достаточно часто, так как количество конфигураций в библиотеке сравнительно небольшое, а число возможных в шахматной партии конфигураций даже при малом количестве фигур технического эндшпиля хотя и ограничено, но весьма велико. В этом случае мы будем иметь дело с аппаратом стремления к библиотечной позиции (см. п. 3.10 и далее).

Приведенная методика позволяет из заложенных в настоящий момент в библиотеку 633 позиций получить до 14000 позиций (без учета симметрий).

3.9. Примеры работы подпрограммы пользования библиотекой эндшпиля. Целью экспериментов с библиотекой технического эндшпиля, кроме проверки эффективности работы подпрограммы пользования справочной эндшпиля при обращении к ней из подпрограммы поиска хода, была также аналитическая проверка оценки заложенной информации. При этом проверялось влияние краевого эффекта на оценку позиций, а также использование аппарата симметрий.

После того, как была закончена шахматная часть работы, т. е. составлена сама библиотека, была разработана специальная подпрограмма, которая расшифровывала решение каждой конфигурации, заложенной в справочную технического эндшпиля (при помощи формул разбиения — см. п. 3.4), и печатала для каждой конфигурации весь массив соответствующих позиций (с их решением). Один такой массив показан на рис. 41.

На выданной машиной диаграмме — заложенная конфигурация; в первой колонке таблицы — номер по порядку; во второй — положение белого короля; в третьей — оценка при ходе белых («+ —» — выиграно у белых, «=» — ничья); в четвертой — ход белых (название фигуры, делающей ход, не выдается; код B9—B9 означает, что указание хода, решающего позицию, не обязательно); в пятой — оценка при ходе черных; в шестой — ход черных.

Такие массивы были розданы для проверки десяткам сильных шахматистов, после чего были устранены некоторые ошибки, неизбежные при составлении такой большой библиотеки и ее программировании.

На рис. 42,*а—в* приведены результаты работы подпрограммы пользования библиотекой. Это решения симметрированных позиций, полученных из одной и той же конфигурации. Поясним подписи на машинных распечатках.

NEW POSITION — новая позиция, представленная подпрограмме для анализа;

WHITE TO PLAY — ход белых;

BLACK TO PLAY — ход черных;

DRAW BY — ничья посредством . . .;

WHITE WINS BY — белые выигрывают ходом . . .;

BLACK WINS BY — черные выигрывают ходом . . . .

3.10. Стремление к библиотечной позиции. Все рассмотренное выше касалось лишь точного совпадения позиций партии (перебора) и библиотечной. Предположим теперь, что позиции, возникшей в партии или в каком-либо варианте перебора, нет в библиотеке эндшпиля, т. е. поиск точного совпадения не привел к положительному результату. Какую же задачу справочного характера ставит перед собой в этом случае шахматный мастер, а следовательно, и программа, моделирующая его мышление?

Х. Р. Капабланка указывал [9], что шахматный мастер не только регистрирует в своих расчетах позиции из библиотеки, чтобы оборвать вариант, но и стремится получить ту или иную выгодную ему библиотечную позицию. Программа должна уметь делать и это.

Пусть конфигурация исходной позиции не найдена среди позиций-символов множеств библиотечных классов. Тогда в библиотеке определяется позиция (или группа позиций), наиболее «близкая» к той, которая получилась в партии или переборе, и организуется *стремление* к этой близкой позиции из исходной. Оно заключается в следующем. В том случае, если рассматриваемая близкая позиция имеет удовлетворяющую нас оценку, в математическое отображение включаются так называемые *планируемые* траектории фигур активной (стремящейся) стороны, т. е. траектории, ходы по которым ведут из исходной позиции к близкой. Активная сторона создает свои зоны. Здесь алгоритм пользования библиотекой действует совместно с алгоритмом поиска хода в оригинальной ситуации. Для того чтобы стремление было организовано, необходимо наличие в математическом отображении траекторий всех несовпадающих фигур пассивной стороны. Важно отметить, что

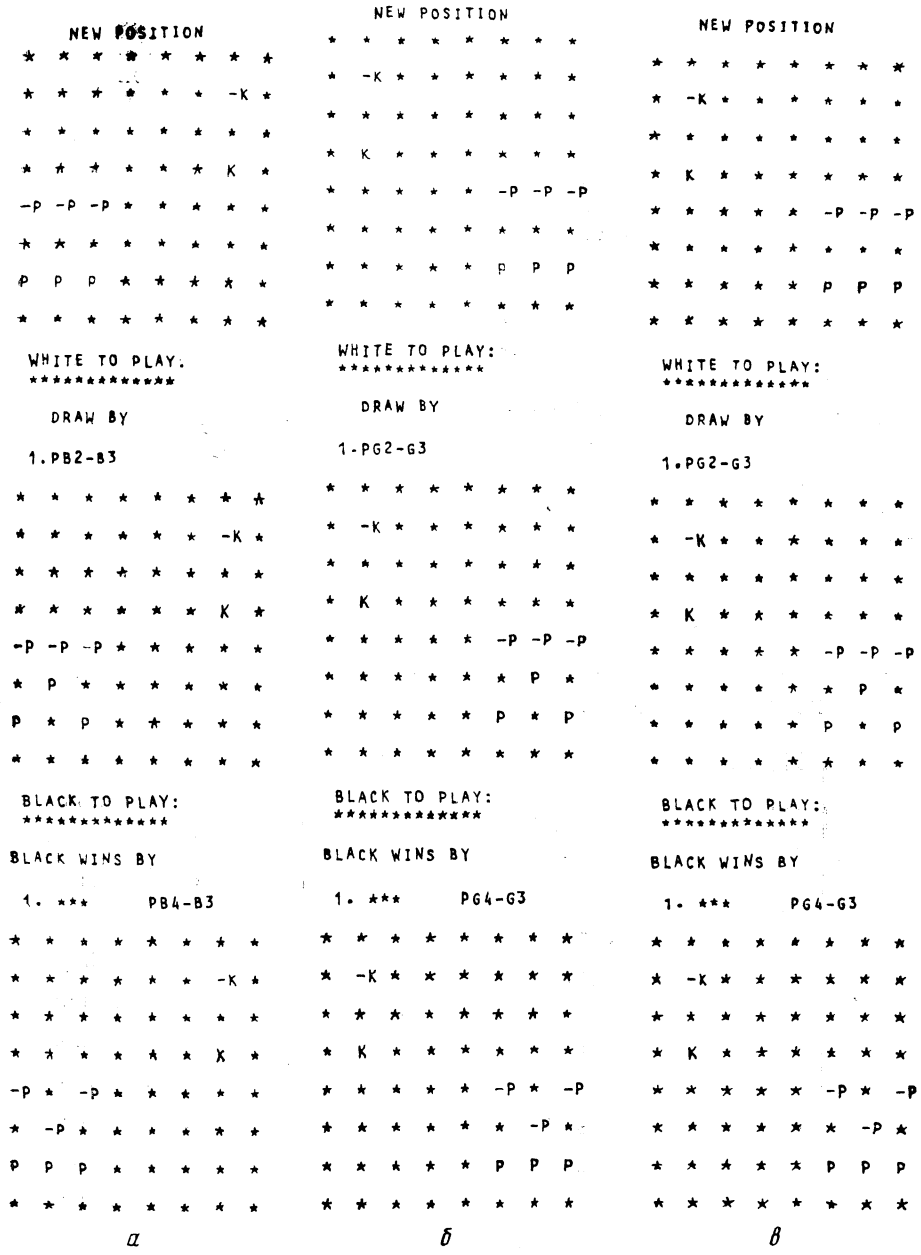

Рис. 42. Образцы результатов работы подпрограммы пользования библиотекой эндшпиля

эти траектории должны быть «антивилочными» в том смысле, что пассивная сторона должна действовать по этим траекториям лишь в крайних случаях. Кроме того, в отличие от основных принципов алгоритма поиска хода в оригинальной ситуации, для препятствия передвижениям фигур пассивной стороны по указанным траекториям никакие зоны не создаются.

Покажем теперь, в чем заключается поиск близких позиций.

3.11. Поиск близких позиций. Прежде чем приступить к описанию поиска близких позиций, определим следующие понятия.

Позиции A и B будем называть позициями *равного материала*, если $U^A{}_i = U^B{}_i$ для любого $i=1, 2, \ldots, 12$, где $U^A{}_i$ и $U^B{}_i$ — элементы шаблонов позиций A и B.

Несовпадением σ_{AB} позиций A и B равного материала назовем

$$\sigma_{AB} = \frac{1}{2}\sum_{i=1}^{64} c_i,$$

где

$$c_i = \begin{cases} 1, & \text{если } a_i \neq b_i; \\ 0, & \text{если } a_i = b_i. \end{cases}$$

Здесь a_i и b_i — принятые в программе коды фигур, расположенных на i-м поле позиций A и B, т. е.

$$a_i = \begin{cases} 11, & \text{если на } i\text{-м поле позиции } A \text{ стоит} & \text{белая пешка} \\ 13, & \text{то же} & \text{белый конь} \\ 14, & \text{»} & \text{белый король} \\ 15, & \text{»} & \text{белая ладья} \\ 16, & \text{»} & \text{белый слон} \\ 17, & \text{»} & \text{белый ферзь} \\ 22, & \text{»} & \text{черная пешка} \\ 23, & \text{»} & \text{черный конь} \\ 24, & \text{»} & \text{черный король} \\ 25, & \text{»} & \text{черная ладья} \\ 26, & \text{»} & \text{черный слон} \\ 27, & \text{»} & \text{черный ферзь} \\ 0, & \text{если } i\text{-е поле позиции } A \text{ пусто;} & \end{cases}$$

i — линейная координата.

Пусть $T_1, T_2, \ldots, T_{\sigma_{AB}}$ — последовательность траекторий, **ходы** по которым переводят позицию A в позицию B*.

Разностью Δ_{AB} позиций A и B равного материала назовем

$$\Delta_{AB} = \sum_{j=1}^{\sigma_{AB}} l_j,$$

где l_j — длина траектории T в полуходах для любого $j=1, 2, \ldots, \sigma_{AB}$.

Позиции A и B равного материала будем называть *близкими*, если $\sigma_{AB} \leqslant \sigma_{\max}$; $\Delta_{AB} \leqslant \Delta_{\max}$, где $\sigma_{\max}$ — максимально допустимое несовпадение (число несовпадающих фигур); $\Delta_{\max}$ — максимально допустимая разность позиций.

В первом приближении было принято, что поиск близких позиций ведется лишь среди позиций равного материала. Это означает, что однозначно определен класс библиотечных позиций-символов конфигураций, материал в которых (с учетом симметрии цвета) в точности совпадает с материалом в исходной позиции (см. рис. 40). Значения величин $\sigma_{\max}$ и $\Delta_{\max}$ являются текущими ограничениями поиска.

Задача состоит в следующем. Имеется исходная позиция A. Требуется найти группу библиотечных позиций $B_1, B_2, \ldots, B_n$, каждая из которых является близкой по отношению к A в указанном выше смысле.

Заметим, что здесь мы не будем анализировать соотношение оценок позиции A и позиций $B_1, B_2, \ldots, B_n$; этот анализ будет проведен ниже. Таким образом, не следует полагать, что к каждой позиции B_i ($i=1, 2, \ldots, n$) впоследствии будет организовано стремление.

Итак, группу позиций $B_1, B_2, \ldots, B_n$ назовем *претендентами на стремление*. Очевидно, что в начале поиска претендентом на стремление является любая позиция из любого множества, описываемого позициями-символами данного класса.

Одной из задач поиска близких позиций является максимальное сокращение числа претендентов на стремление без использования внушительного аппарата нахождения траекторий. Были найдены некоторые ограничения (назовем их *фильтрами*), благодаря которым удается «забраковать» бóльшую часть множеств библиотечных позиций с точки зрения наличия в этих множествах позиций, близких к исходной, забраковать уже на уровне соответствующих конфигураций. Что же это за фильтры?

3.12. Фильтр по относительному расположению пешек. Этот фильтр действует лишь при наличии в исходной (а следовательно,

* В дальнейшем будем говорить (для простоты), что траектории ведут из позиции в позицию, подразумевая под этим, естественно, ходы по траекториям.

и в библиотечных) позиции двух или более пешек любого цвета.

Имеем исходную позицию A и позицию C — символ множества позиций определенной конфигурации. Образуем из позиции A *шаблон пешек*. Для этого условно «спроектируем» расположение всех пешек позиции на первую горизонталь и присвоим восьми полям первой горизонтали значения

$$SH_i^P = \begin{cases} 0, & \text{если нет пешек на } i\text{-й вертикали;} \\ 10p^w{}_i + p^b{}_i, & \text{если на } i\text{-й вертикали есть пешки (пешка),} \end{cases}$$

где $p^w{}_i$ — количество белых пешек на i-й вертикали; $p^b{}_i$ — количество черных пешек на i-й вертикали; $i=1, 2, \ldots, 8$.

Получили массив из восьми элементов SH^P. Далее выделим из этого массива ненулевую часть следующим образом. Отбросим слева нули так, чтобы первым элементом был $SH^P{}_i \neq 0$, а справа так, чтобы последним элементом был $SH^P{}_j \neq 0$ $(j \geqslant i)$. Полученный массив SH^A длиной $l_A = j - i + 1$ назовем шаблоном пешек позиции A.

Аналогичным способом получим шаблон пешек SH^C позиции C длиной l_C.

Докажем следующее утверждение. Не существует последовательности траекторий, переводящих позицию A в любую позицию множества, характерного для C, если а) $l_A \neq l_C$ или б) $l_A = l_C$, но существует хотя бы один номер i из набора $i=1, 2, \ldots, l_A$, такой, что $SH^A{}_i \neq SH^C{}_i$.

Доказательство. Пусть C_1 — одна из позиций множества, характеризующегося позицией-символом C.

Поскольку позиции A и C — равного материала, то и позиции A и C_1 — также равного материала.

Пусть выполнено одно из условий: а) или б).

Предположим противное, т. е. что существует последовательность траекторий, переводящих A в C_1. Тогда, раз выполнено хотя бы одно из указанных условий, то в этой последовательности обязательно есть хотя бы одна траектория, меняющая вертикаль хотя бы одной пешки. Это следует из того, что выполнение одного из условий а) или б) означает несовпадение шаблонов пешек. Но, как известно, пешка может сменить вертикаль лишь посредством взятия (взятий). А это противоречит тому, что A и C_1 — равного материала. И так для любого C_1 из C. Пришли к противоречию. Следовательно, вышеуказанное утверждение доказано.

Значит, если шаблоны SH^A и SH^C не совпадают полностью, то все позиции множества, характерного для C, исключаются из списка претендентов на стремление.

Следует напомнить, что все проведенные рассуждения справедливы с точностью до симметрий. Это означает, что если выполнено условие б), то необходимо из позиции C образовать фланговосимметрированный шаблон пешек $SH^C_{\text{фл}}$ по формуле $SH^C_{i(\text{фл})} = SH^C_{l_C - i + 1}$ $(i = 1, 2, \ldots, l_C)$ и проверить выполнение условия б) для SH^A и $SH^C_{\text{фл}}$.

Если теперь условие б) не выполняется, т. е. $SH^A = SH^C{}_{\text{фл}}$ для любого $i=1, 2, \ldots, l_A$, то C заносится в список конфигураций-претендентов с одновременным запоминанием информации о неизбежности флангового симметрирования.

Кроме того, если материал белых и черных в данном классе одинаков (в шахматном смысле), то необходимо учесть возможность симметрии цвета для SH^C по формуле

$$SH^C_{i(\text{цв})} = \begin{cases} 0, & \text{если } SH^C_i = 0; \\ 10p^b_i + p^w_i, & \text{если } SH^C_i = 10p^w_i + p^b_i. \end{cases}$$

Фильтр по относительному расположению пешек позволяет сильно сократить список претендентов на стремление.

3.13. Фильтр «одноцвет — разноцвет». Этот фильтр действует лишь при наличии в исходной позиции пары слонов противоположного цвета (по одному у каждой стороны).

Дана исходная позиция A. Двумерные координаты белого слона x_w, y_w; черного — x_b, y_b.

Известно, что

$$x + y \begin{cases} \text{четно, для черного поля шахматной доски;} \\ \text{нечетно для белого поля шахматной доски,} \end{cases}$$

если x и y — двумерные координаты определенного поля доски. Следовательно,

$$Z_A = x_w + y_w + x_b + y_b \begin{cases} \text{четно при одноцветных слонах;} \\ \text{нечетно при разноцветных слонах.} \end{cases}$$

Очевидно, не существует последовательности траекторий, переводящих позицию A в любую позицию множества, характерного для C, если Z_A и Z_C имеют разную четность, т. е. $Z = Z_A + Z_C$ — нечетно. Здесь следует отметить, что при работе этого фильтра не требуется учета симметрий, так как любая комбинация симметрированных преобразований над позицией C не изменит четности величины Z_C, а следовательно, и четности величины Z.

Таким образом, сформирован первоначально «отфильтрованный» список конфигураций-претендентов $C_1, C_2, \ldots, C_m$, т. е. позиции — претенденты на стремление могут находиться только в множествах, характеризующихся этими конфигурациями.

3.14. Фильтры внутри множества позиций. Приступим теперь непосредственно к формированию списка позиций — претендентов на стремление.

Имеем исходную позицию A и очередную конфигурацию-претендент C_i. Очевидно, не все позиции множества, характеризующегося C_i, будут претендентами на стремление. Укажем ограничения (фильтры), помогающие уменьшить количество этих позиций.

Во всех нижеследующих рассуждениях мы не будем упоминать аппарат симметрий. На самом же деле работа этого аппарата существенна на всех этапах алгоритма поиска близких позиций.

Итак, очередным сужением круга претендентов на стремление является выбор лишь тех позиций из всего их множества, у которых вертикали всех пешек в точности совпадают с вертикалями пешек исходной позиции A. Только эти позиции, согласно доказанному в п. 3.12 утверждению, являются претендентами. Но и это еще не все. Необходимо также, чтобы координаты $y^A{}_j$ всех белых пешек позиции A удовлетворяли условию

$$y^A{}_j \leqslant y^C{}_j,$$

а для черных пешек — условию

$$y^A{}_j \geqslant y^C{}_j.$$

Здесь $y^C{}_j$ — вторые координаты (вертикали) соответствующих пешек в рассматриваемой позиции из C_i. Иначе говоря, ни одна пешка исходной позиции не должна быть «впереди» соответствующей пешки из библиотечной позиции, так как в противном случае пришлось бы искать траектории, перемещающие пешки назад. Но пешки назад не ходят и, следовательно, таких траекторий не существует.

Далее, если в позиции-претенденте есть слоны, то их белопольность или чернопольность должна соответствовать позиции A.

Наконец, все фильтры внутри множеств пройдены, и мы получаем первоначальный список позиций-претендентов на стремление.

Необходимо отметить в заключение, что в программе, реализующей данный алгоритм поиска близких позиций, не запоминаются текущие позиции-претенденты, а фиксируются лишь местоположения в программе формул разбиения, воспроизводящих эти позиции по символам соответствующих множеств.

3.15. Получение группы близких позиций. Когда список позиций-претендентов стал небольшим, можно приступить к непосредственной проверке «близости» каждой из них к исходной позиции.

Первоначально проверяется, не превышено ли максимально допустимое несовпадение (σ_{max}). Список претендентов еще более уменьшается. Теперь, когда количество несовпадающих фигур не превышает установленного лимита, необходимо перейти к использованию аппарата получения траекторий. После нахождения траекторий несовпадающих фигур, т. е. именно тех траекторий, которые переводят близкую позицию в позицию исходную, можно, проверив, не превышена ли максимально допустимая разность позиций (Δ_{max}), принимать решение о включении претендентов в искомую группу близких позиций. Только теперь получена окончательная последовательность позиций $B_1, B_2, \ldots, B_n$, близких к исходной позиции A.

Отметим, что значения σ_{max} и Δ_{max} могут варьироваться в зависимости от наличия ресурсов времени и памяти. По нашему мнению, точно так же действует и шахматный мастер. В условиях цейтнота или перегруженной памяти большие значения σ_{max} и Δ_{max} не позволит себе и чемпион мира.

Блок-схема алгоритма поиска близких позиций приведена на рис. 43.

3.16. Реализация стремления. Антистремление. Поясним теперь еще раз, что же понимается под стремлением из исходной позиции A к близкой библиотечной позиции B. Организовать такое стремление означает включить с соответствующими приоритетами в математическое отображение траектории, ведущие из позиции A к позиции B, оценка и решающие ходы которой при различных очередях хода нам известны из библиотеки технического эндшпиля.

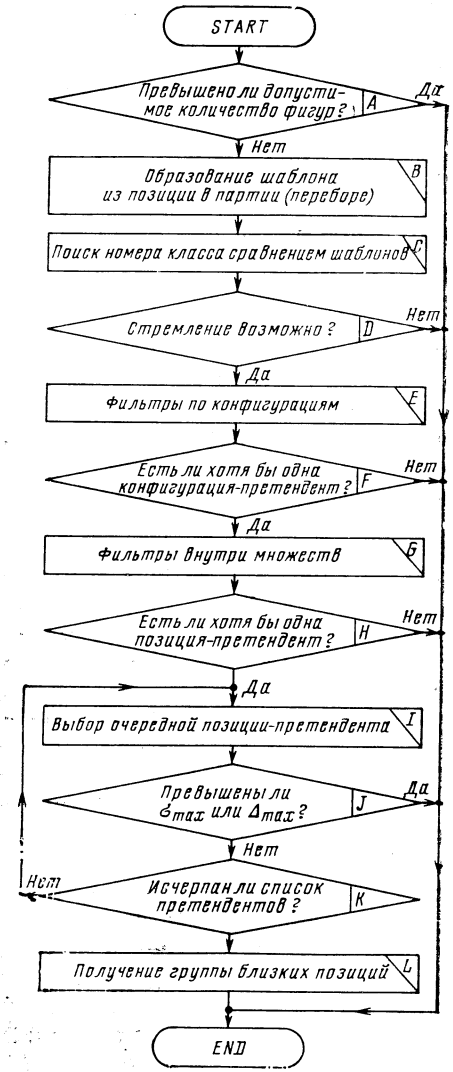

Рис. 43. Блок-схема алгоритма поиска близких позиций для стремления

Реализация метода, основанного на стремлении к библиотечной позиции, может привести к направленному формированию математического отображения и дерева перебора. Но как только совпадение будет достигнуто, в действие вступает справочный метод, основанный на точном совпадении позиций, оценка становится известной и вариант обрывается.

Введем теперь понятие антистремления.

Предположим, что в исходной позиции A в белом узле дерева перебора (т. е. ход белых) оценка — выиграно у белых. Среди близких позиций есть позиция B, в которой оценка при ходе белых — ничья. Пусть также в математическом отображении позиции по каким-либо причинам имеются траектории белых фигур, ведущие из A в B. Тогда *антистремлением* назовем искусственное снижение приоритетов ходов по этим траекториям. Таким образом, антистремление позволяет, если можно так выразиться, избежать ловушек.

3.17. Обращение из подпрограммы поиска хода. Рассмотрим теперь, как происходит обращение к подпрограмме стремления из подпрограммы поиска хода в оригинальной ситуации. Если в эндшпиле в каждом узле дерева перебора проверять совпадение позиции с позицией из библиотеки, то на это уйдет немного времени — это процедура справочного характера. Но если проверять позиции во всех узлах дерева на стремление, это будет нарушением принятого алгоритма включения зон. Прежде чем стремиться, следует выяснить, что это стремление может дать дополнительно, поскольку всякое стремление связано с включением зон и расширением дерева перебора. Поэтому для стремления принят тот же порядок, что при включении зон, т. е. позиции проверяются на стремление лишь при подъеме по варианту.

Напомним, что если в математическом отображении позиции нет хотя бы одной из планируемых траекторий пассивной стороны, то стремление к такой позиции не организуется и позиция «вычеркивается» из списка претендентов на стремление.

3.18. Стремление и точное совпадение. Выскажем теперь некоторые соображения об общности и различии справочного метода и метода стремления.

Из вышеизложенного ясно, что поиск точного совпадения позиций является частным случаем стремления к близкой позиции. Действительно, позицию в библиотеке, в точности совпадающую с исходной, можно считать в общем случае близкой позицией.

Пусть дана исходная позиция A и некоторая библиотечная позиция B, являющаяся позицией равного с A материала. Пусть также установлено, что $\sigma_{AB}=0$; $\Delta_{AB}=0$.

Тогда, во-первых, позиция B в точности совпадает с позицией A и, во-вторых, согласно определению близких позиций, позиция B является близкой к A.

Следовательно, при нулевых несовпадении и разности между двумя близкими позициями эти позиции в точности совпадают. В этом — общность справочного метода и метода стремления.

Различие же между этими двумя методами при использовании опыта прошлого состоит лишь в том, что справочный метод, являясь частным случаем метода стремления, не требует применения алгоритма поиска хода в оригинальной ситуации, а точнее, во-первых, не требует использования аппарата получения траекторий и, во-вторых, не связан с некоторым расширением дерева перебора вариантов. Именно поэтому использование справочного метода является более простым, т. е. требует меньшего расхода ресурсов ЭВМ. С этим связан и избранный порядок изложения использования этих методов при составлении библиотеки технического эндшпиля и алгоритмов пользования этой библиотекой.

3.19. Возможность широкого использования библиотечных правил в программе «Пионер». Укажем в заключение на такую часть библиотеки эндшпиля, использование которой не связано со сравнением (полным или частичным) позиции варианта перебора со встречавшимися ранее. Речь идет о так называемой *библиотеке правил.* В процессе развития теории принятия решения в любой области выработано множество методов, позволяющих оценивать создавшуюся ситуацию без дальнейшего формирования дерева перебора возможностей, а лишь на основании некоторых критериев, правил, имеющих характер строго доказанных теорем. Не являются исключением и шахматы. Чем больше таких правил знает мастер, тем больше вероятность скорейшего обрыва вариантов дерева перебора на основании этих правил.

При составлении любой программы, базирующейся на построении дерева перебора вариантов и минимаксной процедуре на этом дереве, перед составителями всегда возникает проблема, связанная с принудительным обрывом варианта дерева перебора по некоторым критериям до достижения предельной длины вариантов. Эта задача впервые решена в программе «Пионер». Здесь речь идет не об обрывах, связанных с целью игры (этим занимается подпрограмма поиска хода в оригинальной ситуации), а об обрывах вариантов по некоторым библиотечным правилам. Очевидно, что окончательные размеры дерева перебора тем меньше, чем большее число вариантов удалось оборвать таким образом. Естественным поэтому является желание, во-первых, выработать и формализовать достаточно большое количество таких критериев для обрывов и, во-вторых, использовать эти критерии в как можно большем числе узлов дерева перебора.

Однако время, требуемое для решения задачи, может сильно возрастать при использовании указанных критериев в большом количестве узлов дерева. Эффективность решения связана, следовательно, с размерами самого дерева перебора вариантов. Поскольку проверка критериев обрыва сопровождается некоторой, пусть даже незначительной (на узел), затратой времени, то становится ясной бессмысленность такой проверки при больших размерах дерева. Здесь речь идет, конечно, не о формальных отсечениях типа α—β-процедуры, а о критериях обрыва, связанных со специ-

фикой задачи, в нашем случае с библиотечными шахматными правилами.

Очевидно, что для большинства существующих шахматных программ, при работе которых возникают (вследствие заложенных принципов полного перебора) громадные деревья вариантов, возможность эффективного использования библиотечных правил сомнительна.

Иная картина наблюдается при составлении программы «Пионер». Здесь мы имеем дело с небольшим, узким (что чрезвычайно важно) деревом перебора — порядка 10^2 узлов. При принятом алгоритме имеется возможность широкого использования библиотечных правил. Необходимо отметить и наглядно просматриваемую здесь обратную связь: именно эффективное использование правил (методов игры в различных стадиях шахматной партии) позволяет еще более сократить дерево перебора вариантов, что не только ускорит решение задачи, но и качественно повлияет на эффективность этого решения.

Таким образом, при решении неточных переборных задач по алгоритму М. Ботвинника проявляется взаимосвязь между небольшим деревом перебора и использованием опыта накопленных знаний о предмете исследования, в данном случае о шахматной игре.

3.20. Критерии обрыва вариантов, основанные на «правиле квадрата». За многовековую историю своего развития шахматная теория выработала огромное количество методов игры, всевозможных приемов, позволяющих сократить расчет вариантов.

При реализации шахматной программы «Пионер» целесообразным для повышения эффективности решения задачи является, как уже отмечалось, использование наиболее существенных из этих методов, формализованных в критерии для обрыва вариантов.

В качестве примера рассмотрим формализацию и расширение так называемого «правила квадрата» для пешечных окончаний. Нет необходимости разъяснять здесь суть шахматного правила квадрата. Оно подробно изложено в многочисленной специальной литературе.

Поскольку программа может играть любым цветом, то мы не будем говорить конкретно о белых или о черных, а лишь о стороне (+) и стороне (—). Примем следующие обозначения:

$Кр_{(+)}$ — король стороны (+);
$П_{(+)}$ — пешка стороны (+);
$Кр_{(-)}$ — король стороны (—);
$П_{(-)}$ — пешка стороны (—).

Абсолютный критерий. Материал: $Кр_{(+)}$, $П_{(+)}$; $Кр_{(-)}$.

1. Если $Кр_{(-)}$ не в квадрате * $П_{(+)}$, то сторона (+) выигрывает.
2. Если $Кр_{(-)}$ в квадрате $П_{(+)}$ и ближе к $П_{(+)}$, чем $Кр_{(+)}$, то — ничья.

* Здесь и далее при определении нахождения короля в квадрате пешки учитывается очередь хода. Например, в одной и той же позиции, в зависимости от очереди хода, король может находиться или не находиться в квадрате пешки (это граничный случай — король на границе квадрата).

Здесь и далее понятие «ближе» означает, что
$|Кр_{(-)}—П_{(+)}| < |Кр_{(+)}—П_{(+)}|$,
где $|Кр_{(-)}—П_{(+)}|$ и $|Кр_{(+)}—П_{(+)}|$ — расстояния в полуходах между соответствующими фигурами. Эти расстояния элементарно вычисляются по координатам фигур. Приведенная формула учитывает очередь хода.

Первый критерий достаточности. Материал: $Кр_{(+)}$, $П_{(+)}$; $Кр_{(-)}$, $П_{(-)}$.

Пусть оба короля находятся в квадратах неприятельских пешек.

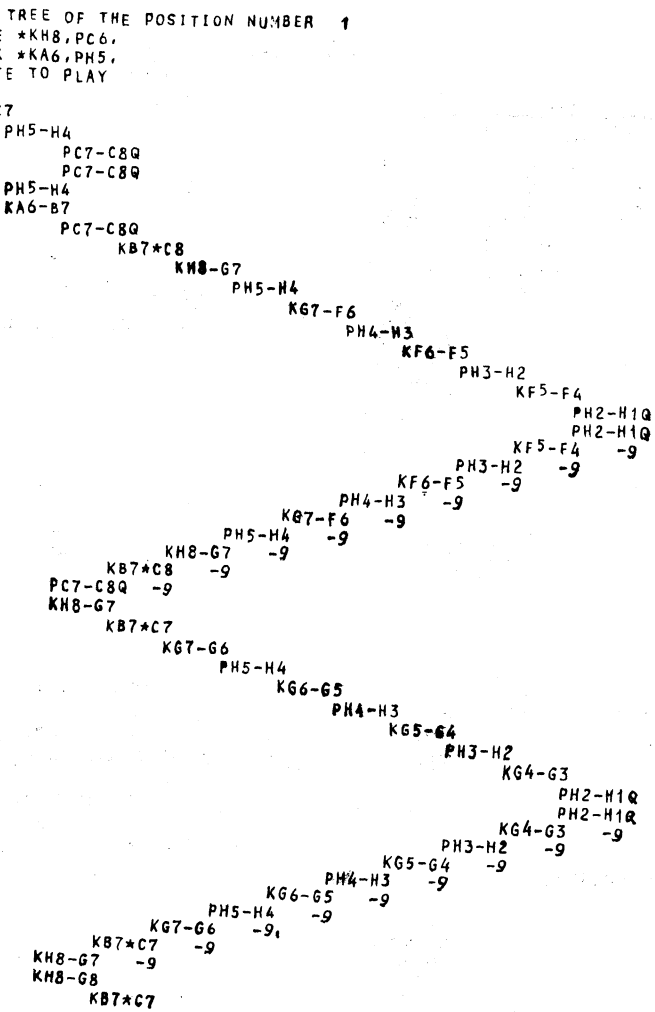

Рис. 44. Распечатка части дерева перебора при решении программой этюда Р. Рети до введения критериев обрыва вариантов, основанных на правиле квадрата

Для того, чтобы одной из сторон была гарантирована ничья, достаточно, чтобы король этой стороны находился ближе к неприятельской пешке, чем его оппонент (к той же пешке). Например, если

$$|Кр_{(+)}—П_{(-)}| < |Кр_{(-)}П_{(+)}|$$

с учетом очереди хода, то стороне (+) ничья гарантирована.

Докажем это. Пусть выполняется приведенное неравенство. В этом случае, если сторона (+) будет делать ходы только королем в направлении неприятельской пешки так, чтобы расстояние между ними уменьшалось с каждым ходом (что возможно вследствие выполнения правила квадрата), то $Кр_{(+)}$ беспрепятственно уничтожит пешку $П_{(-)}$ до или в момент ее превращения. И тогда стороне (+) ничья гарантирована.

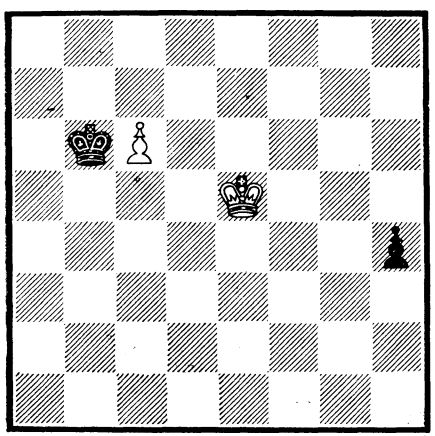

Рис. 45. Позиция из дерева перебора решения программой «Пионер» этюда Р. Рети (Белые : Кр h 8, п. с. 6; Черные : Кра6, п. h5. Ничья)

Второй критерий достаточности. Материал: $Кр_{(+)}$, $П_{(+)}$; $Кр_{(-)}$, $П_{(-)}$.

Пусть

— $Кр_{(-)}$ находится в квадрате $П_{(+)}$;

— $Кр_{(+)}$ находится не дальше от пешки $П_{(+)}$ и от горизонтали ее превращения, чем $Кр_{(-)}$;

— $П_{(-)}$ находится не на вертикалях от $Кр_{(+)}$ до $П_{(+)}$ включительно;

— $Кр_{(+)}$ и $П_{(+)}$ — не на ладейных и коневых вертикалях.

Тогда: а) если $П_{(+)}$ проходит в ферзи на один полуход раньше, чем $П_{(-)}$, то ничья гарантирована стороне (+); б) если $П_{(+)}$ проходит в ферзи на один полуход позже, чем $П_{(-)}$, стороне (+) гарантирована ничья (если только $П_{(-)}$ не превращается в ферзя с шахом и если новоявленный $Ф_{(+)}$ не уничтожается посредством «вилочного» шаха). Простое доказательство справедливости этого критерия громоздко по объему, поэтому мы его не приводим.

Следует отметить, что понятия «находится в квадрате», «находится не дальше», «находится не на вертикалях», «проходит в ферзи», «раньше», «превращается в ферзя с шахом», «уничтожается посредством вилочного шаха», и т. п. в программе четко формализованы и определяются с учетом очереди хода.

В результате введения в программу «Пионер» критериев (обрыва вариантов), основанных на расширенных правилах квадрата, дерево решения этюда Р. Рети (см. рис. 13) приняло «человеческие» очертания. До введения в программу этих критериев «Пионер» продолжал бессмысленный перебор даже в тех позициях, где шахматист давно оборвал бы вариант (см. рис. 44).

Необходимо отметить, что программа в некоторых случаях обрывает вариант после хода черных, но конечная оценка учитывает предполагаемый ответ белых. Поясним это на примере одной из позиций дерева перебора, построенного «Пионером» при решении этюда Р. Рети.

В позиции на рис. 45 после 3. ... Крb6 : c6 шахматист оборвал бы вариант из-за ответа 4. Крe5 — f4, программе же даже не надо совершать этот ход, так как абсолютный критерий уже действует. Точно так же, в ответ на 3. ... h4—h3 — ничья по второму критерию достаточности, а шахматист видит 4. Крe5 — d6.

3.21. Заключение. Таким образом, при создании для ЭВМ шахматной программы, моделирующей мышление шахматного мастера, одной из важнейших является задача создания информационно-справочной системы «Опыт прошлого», а также разработка алгоритмов, позволяющих программе эффективно использовать этот опыт.

Есть основания полагать, что использование методов справочного и стремления, а также библиотечного обрыва вариантов аналогично тому, как это делает шахматный мастер (и программа «Пионер»), может найти применение и в других практических областях управления, где, подобно шахматам, возникают неточные задачи переборного типа.

ПРИЛОЖЕНИЕ 4

АССОЦИАТИВНАЯ БИБЛИОТЕКА ФРАГМЕНТОВ

А. И. Резницкий, А. Д. Юдин

Использование справочного метода, рассмотренного в приложении 3, не может быть эффективным в середине игры, поскольку в этой стадии партии одинаковые позиции возникают крайне редко. Более подходящим для миттельшпиля представляется метод, способствующий принятию решения о выборе хода в создавшейся позиции по аналогии (ассоциации) с ранее встречавшимися.

Метод получения информации, основанный на ассоциации по сходству анализируемой позиции и позиции, сведения о которой занесены в библиотеку, был назван *ассоциативным*, а сама библиотека, хранящая признаки сходства позиции, — *ассоциативной*.

Перспективность использования шахматными алгоритмами опыта прошлого по ассоциативному методу была отмечена еще К. Шенноном [2]. Он приводил в пример шахматного мастера, который «знает сотни, а может быть, и тысячи стандартных позиций, привычных комбинаций и типовых маневров, которые встречаются неоднократно в партиях. Имеются, например, стандартные жертвы коня на f7 или слона на h7, стандартные маты, например мат Филидора, маневры, связанные с вилками, превращениями и т. д. В данной позиции он усматривает много сходства со знакомыми ему случаями, и это направляет его мысль на исследование тех вариантов, в которых вероятность успеха наибольшая».

В попытке шахматиста применить в игре опыт прошлого к настоящему, найдя между ними некоторое сходство, и проявляется ассоциативность его мышления.