М.М. Ботвинник

О КИБЕРНЕТИЧЕСКОЙ ЦЕЛИ ИГРЫ

М. М. Ботвинник

О КИБЕРНЕТИЧЕСКОЙ ЦЕЛИ ИГРЫ

Москва · „Советское радио“ · 1975

6ФО.1
Б 86
УДК 007→518.92

Ботвинник М. М. О кибернетической цели игры. М., «Сов. радио», 1975, 88 с.

Рассматриваются общие вопросы работы систем управления, их классификация, требования, которым должен отвечать алгоритм работы этих систем. Отмечается особое значение кибернетической цели игры для составления алгоритма и эффективности работы систем управления. Приводится алгоритм игры в шахматы как пример, поясняющий общие вопросы теории. М. М. Ботвинник — доктор технических наук, экс-чемпион мира по шахматам. Последнее время активно работает в области машинной игры в шахматы.

Книга рассчитана на широкий круг читателей.

Рис. 19, табл. 1.

Редакция кибернетической литературы

Б $\frac{30501\text{-}034}{046(01)\text{-}75}$ 91-74

От редактора

Прошло семь лет с момента выхода в свет книги М. Ботвинника «Алгоритм игры в шахматы». Это не малый срок в жизни человека, и читатель, вероятно, хочет узнать, что произошло за эти годы, приблизился ли Ботвинник к достижению своей цели? Произошло немало событий. Ботвинник переступил через рубеж шестидесяти лет, перестал выступать на шахматной арене, посвятил себя полностью работе по созданию искусственного шахматиста-гроссмейстера, создал и возглавил группу, работающую над составлением шахматной программы для ЭВМ, довел алгоритм игры в шахматы почти до конца (работа над алгоритмом будет закончена только после испытания и возможной корректировки программы). Идеи, изложенные Ботвинником семь лет тому назад, не только получили свое развитие и уточнение, но и подверглись в некоторой своей части значительной переработке. Настоящая книга не является переизданием предыдущей. Однако можно с уверенностью сказать, что и она вызовет большой интерес.

Машинная игра в шахматы — это очень интересно, но машинная игра в силу гроссмейстера — это потрясающе интересно. А именно такова цель Ботвинника, и есть основания надеяться, что она будет достигнута. Выше было перечислено то, что произошло за истекшие годы. Скажем теперь, что за это время не произошло. Ботвинник не потерял своей энергии, не утратил замечательной способности анализировать свою собственную игру, свое собственное шахматное мышление. Новая книга Ботвинника так же доступна для широкого круга читателей, как предыдущая. Она безусловно привлечет к себе внимание и многих ученых: математиков, специалистов по кибернетике, психологов, экономистов, работающих в области автоматизации процессов планирования. И, конечно, ее пожелает прочесть каждый шахматист.

Как уже было сказано, в книге рассматривается вопрос о машинной игре в шахматы. Остановимся прежде всего на том, о каких машинах идет речь и каким образом они могут играть в шахматы. Имеются в виду современные электронные вычислительные машины дискретного действия, так называемые ЭВМ, которые относятся к числу программно управляемых. Применение электронной техники делает их очень быстродействующими, однако основная их черта — автоматическое управление процессами переработки информации — вовсе не связана с этой техникой. Такие машины представляют собой сложные автоматы, состоящие из ряда связанных между собой устройств: устройства управления, запоминающего устройства (называемого также памятью), операционного устройства, устройства ввода информации и устройства выдачи результатов. Информация может быть введена в программно управляемую машину и фиксирована в ее памяти в алфавитно-цифровом виде (в виде слов и чисел, которые представлены, понятно, в машинном коде).

Информацию, вводимую в машину, можно, грубо говоря, разделить на управляющую и перерабатываемую. Управляющей информацией является программа работы машины, состоящая из некото-

рого числа так называемых команд. Устройство управления в каждом такте работы прочитывает в программе, введенной в память, команды одну за другой и заставляет все устройства машины выполнять операции, предусмотренные в этих командах. Само устройство управления также выполняет некоторые из таких операций, например, производит поиск очередной команды или их видоизменение и т. п. После получения искомого результата, выполняя соответствующую команду, машина выдает его в том или в ином виде. Современные машины, кроме ряда других устройств выдачи, имеют электрическую пишущую машинку, которая может также служить устройством ввода. Если человек-оператор напечатает что-либо на этой машинке, набранный на ее клавиатуре текст вводится в память машины. В свою очередь, при наличии в программе соответствующей команды, машинка автоматически печатает полученные результаты. Обычно такая машинка использует двухцветную ленту. Текст, вводимый в машину, печатается одним цветом (например, черным), а выводимый из машины — другим (например, красным). Таким образом, диалог человека-оператора и машины фиксируется на бумаге в виде дословного протокола. Машина, конечно, располагает и устройствами ввода и вывода больших объемов информации, но для игры в шахматы большое удобство представляет именно такая пишущая машинка (еще лучше дисплей).

Гибкость работы программно управляемой машины достигается тем, что устройство управления может изменять порядок выбора команд в зависимости от получаемых результатов, а также тем, что управляющая информация, как и перерабатываемая, представлена в алфавитно-цифровом виде и может подвергаться переработке. Современные программы составляются так, что, кроме выполнения операций над перерабатываемой информацией, в них предусмотрено также видоизменение входящих в них команд и выполнение их в новом виде, так что по короткой программе машина может выполнять очень длинные последовательности операций.

Машинная игра в шахматы протекает следующим образом. Заранее разработанную программу вводят в машину через устройство ввода (описанная выше пишущая машинка непригодна для этой цели). Затем в память машины вводят (в качестве перерабатываемой информации) описание начальной позиции на доске Устанавливают право первого хода, и эту информацию тоже вводят в память машины. Если первый ход принадлежит машине, то, выполняя программу, она вычисляет его и выдает на пишущую машинку. Одновременно машина вычисляет новую позицию и сохраняет ее в своей памяти вместо предыдущей. После этого машина «ждет» (для чего в программе предусматривают «цикл» из команд, выполнение которых не затрагивает ни программы, ни перерабатываемой информации, из которого машина может выйти только в результате так называемого прерывания, возникающего, в частности, и при вводе информации через пишущую машинку).

Шахматист, сражающийся с машиной, воспроизводит на доске выданный ею ход и делает ответный ход, запись которого вводят в машину через пишущую машинку. Если первый ход принадлежит человеку, то, получив об этом информацию, машина «ждет» до тех пор, пока в нее не введут информацию о ходе ее противника Дальнейшее течение игры одинаково как при первом ходе машины, так и при первом ходе человека.

Возможна и игра одной машины против другой и даже игра в одной и той же машине одной программы против другой. Но на этих, вообще говоря, интересных вопросах мы останавливаться не будем.

Как уже было сказано, программа представляет собой ряд команд, каждая из которых для каждого варианта исходных данных определяет однозначно операции, выполняемые устройствами машины, в том числе и выбор следующей команды. Таким образом, машина является исполнителем предписания, заданного в виде программы. Другими словами, программа — это предписание, определяющее действия машины и изложенное на машинном языке. Предписание, которое для каждого конкретного исходного данного определяет процесс переработки информации, получило в науке название алгоритма. Можно сказать, что, играя в шахматы, программно управляемая машина выполняет алгоритм шахматной игры.

Алгоритм шахматной игры вначале составляют не на машинном, а на каком-либо другом, более удобном для человека, точном (формальном) языке, например на языке математических описаний. После этого проблема сводится к переводу с одного языка на другой. Перевод алгоритма на машинный язык получил название программирования.

Первоначальное составление алгоритма представляет собой творческую работу. Человек, создающий алгоритм, приводит в действие все свои творческие способности: воображение, предвидение, художественное чувство (да! алгоритм должен быть не только эффективным, но и красивым, изящным), тонкий научный анализ, изобретательность и многое другое. Однако и программирование — в значительной части творческая работа. При программировании приходится решать каждый раз заново много сложных вопросов, например: определение общей организации программы, распределение работы по выполнению алгоритма между различными ее устройствами (это особенно важно и сложно в современных машинах, устройства которых могут работать параллельно), размещение исходных данных и результатов в памяти, контроль за правильностью получаемых результатов (ибо при эксплуатации машин на всех этапах этой работы могут возникать случайные ошибки).

Итак, чтобы заставить машину играть в шахматы, нужно сначала создать алгоритм шахматной игры. Для этого нужно, во-первых, проанализировать шахматную игру, убедиться что такой алгоритм возможен, и, во-вторых, составить его. Третьим этапом будет программирование.

Возникает вопрос, нельзя ли позаимствовать алгоритм игры в шахматы у кого-нибудь из видных шахматистов? Этот вопрос тесно связан с темой данной книги Ботвинника, и к нему мы вернемся немного позже.

Возможен и другой способ игры машины в шахматы. Можно разработать алгоритм самообучения машины игре в шахматы, по которому на основании ряда игр машина сама для себя вырабатывает алгоритм шахматной игры и при дальнейших играх его совершенствует. Проблема создания программ для самообучения машин весьма интересна и очень трудна. Для ее решения нужно представить в виде алгоритма процесс творческой деятельности. Пока что методика создания алгоритмов самообучения находится на самой начальной стадии своего развития.

В данной книге говорится об игре машины по заранее составленной программе, являющейся готовым алгоритмом шахматной игры. В пользу создания такой программы (и против самообучающейся программы), кроме неизученности проблемы создания алгоритмов самообучения, говорит еще и то обстоятельство, что люди в течение тысячелетий играют в шахматы и многому успели научиться. По-видимому, машина очень нескоро накопит такой же по своему объему опыт. Поэтому, если мы хотим побыстрее увидеть совершенную игру машин в шахматы, целесообразно составить алгоритм шахматной игры, используя накопленный человечеством опыт. Но этот путь требует, чтобы разработкой алгоритма шахматной игры занялся человек, изучивший этот опыт. А Ботвинник именно такой человек.

Многие гроссмейстеры играют в шахматы хорошо и даже отлично, но ни один из них не знает алгоритма точной игры в шахматы. Если их методы алгоритмизировать, то получатся алгоритмы более или менее хорошей игры в шахматы. Существующие сегодня программы игры машин в шахматы могут быть названы алгоритмами лишь более или менее удовлетворительной игры. Но возможен ли алгоритм точной игры, или, как говорят математики, существует ли такой алгоритм? Прежде всего, что это такое? Выше было сказано, что алгоритм игры в шахматы — это алгоритм, который при любой заданной шахматной позиции позволяет вычислить ход. Под алгоритмом точной игры в шахматы, мы будем понимать алгоритм, который, если во время игры руководствоваться только им, ведет неукоснительно к достижению цели.

Займемся теперь рассмотрением шахмат с точки зрения возможности составления алгоритма точной игры и попутно выясним, какова должна быть при этом ее цель.

Покажем, что в ходе шахматной игры могут встречаться ситуации только трех видов: 1) обеспеченный (при правильной игре игрока) выигрыш, 2) бесспорный (при правильной игре противника) проигрыш, 3) ничья (при правильной игре обоих игроков). Одновременно покажем, что существует алгоритм точной и наилучшей игры в шахматы, пользуясь которым можно при ситуации обеспеченного выигрыша сделать мат за наименьшее число ходов, при ничейной ситуации не проиграть, а в случае отступления противника от алгоритма наилучшей игры перейти к ситуации обеспеченного выигрыша, при ситуации бесспорного проигрыша — проиграть за наибольшее число ходов или при ошибке противника перейти к самой лучшей из ставших возможными ситуации. То, что сказано в предыдущей фразе о возможностях алгоритма точной (и наилучшей) игры в шахматы, именно и должно быть принято за цель шахматной игры, которую нужно иметь в виду, разрабатывая алгоритм точной игры.

Сразу подчеркнем, что упомянутый алгоритм существует лишь потенциально, что он до сих пор не составлен и в том виде, в котором мы его опишем, практически невыполним из-за того, что требует осуществления астрономического числа машинных операций. Тем не менее факт его существования интересен, он обнадеживает, позволяет предполагать, что когда-нибудь будет реально построен эффективный алгоритм точной игры в шахматы. Какие же к этому основания? Оказывается, такие основания есть. Во-первых, математический факт, что для каждого алгоритма существует бесконечное количество эквивалентных ему алгоритмов, приводящих к получению тождественных результатов, хотя и различными путями. Во-вторых,

практически невероятно, чтобы из бесконечного числа алгоритмов, решающих нашу задачу, описываемый ниже алгоритм предусматривал самый короткий путь решения. Можно даже доказать (это доказательство не может быть здесь приведено), что наш алгоритм предусматривает вовсе не самый короткий путь получения хода. Наконец, в-третьих, существует путь, идя которым можно, отправляясь от описанного ниже алгоритма точной игры, получать эквивалентные ему «лучшие» алгоритмы. Что это за путь будет сказано несколько позже.

Прежде всего, убедимся в том, что и для белой и для черной стороны вопрос о том, как следует играть, может решаться одинаково. Для этого покажем, что реальная игра в шахматы эквивалентна игре, которую мы назовем белой игрой. Белую игру можно осуществить, если каждому из двух сражающихся шахматистов дать отдельную доску, на которой он играет белыми, и, привлечь помощника, который после каждого хода, совершаемого на одной из досок белыми, перемещает на второй доске черную фигуру, одноименную с переместившейся белой и симметрично расположенную с этой белой фигурой относительно «вертикальной» оси (т. е. оси, делящей доску на правую и левую половины).

Итак, для простоты, мы свели задачу исследования шахматной игры к задаче исследования белой игры. При белой игре шахматная партия представляет собой последовательность позиций, в каждой из которых ход принадлежит белым. Если все позиции перенумеровать в порядке их возникновения с помощью целых чисел 1 (начальная позиция), 2, 3, ..., то позиции 1, 3, 5, ... возникают перед первым из шахматистов, а позиции 2, 4, 6, ... — перед вторым.

Ход можно рассматривать как совершаемое по правилам игры преобразование позиции в следующую за ней. Игра, завершающаяся выигрышем одной из сторон, кончается позицией, в которой белым сделан мат. Позиции, в которых сделан мат черным, в белой игре вовсе отсутствуют. Итак, мы убедились в том, что, изучая ситуации на доске, мы можем ограничиться ситуациями, возникающими для белых.

Теперь покажем, что количество различных между собой позиций на шахматной доске конечно. Под позицией мы будем понимать расположение фигур на доске, снабженное, если это может быть нужным для выбора хода, некоторой дополнительной информацией Именно информацией о возможности взятия пешки на проходе и о возможности рокировки для каждого из играющих. Сразу подчеркнем, что нас интересует лишь сам факт конечности множества шахматных позиций и не интересует их число. Нужный нам результат можно получить следующим образом.

Введем следующий способ кодирования произвольной шахматной позиции. Шахматную доску будем изображать в виде последовательности 64 символов, каждый из которых соответствует одной из клеток шахматной доски. При этом, если некоторая клетка является пустой, то соответствующий ей символ должен быть нулем, а если на этой клетке расположена некоторая фигура (безразлично какая), то указанный символ должен быть единицей. Таким образом, шахматная доска представлена в виде строки из 64 нулей и единиц. Нужно теперь только добавить информацию о том, что за фигуры стоят на местах, изображенных в виде единиц, и двигались

ли короли и ладьи, а также имеется ли возможность взятия какой-либо пешки на проходе. Примем следующие обозначения:

белая пешка	— 0000	черный слон	— 0111
черная пешка	— 0001	белая ладья	— 1000
белая пешка, допускающая взятие на проходе	—0010	черная ладья	— 1001
		белая ладья, которая не двигалась	— 1010
черная пешка, допускающая взятие на проходе	— 0011	черная ладья, которая не двигалась	—1011
		белый ферзь	— 1100
белый конь	— 0100	черный ферзь	— 1101
черный конь	— 0101	белый король	— 1110
белый слон	— 0110	черный король	— 1111

Дальнейшее построение кода позиции выполним так. Будем двигаться по строке, изображающей шахматную доску, слева направо и каждый раз, встречая единицу, будем справа от имеющейся строки приписывать четверку цифр, обозначающую шахматную фигуру. Процесс закончим, достигнув самой правой цифры в изображении доски (т. е. 64-й, если считать слева направо). К полученной строке, если это не вытекает из расположения фигур, добавим справа одну цифру, характеризующую возможность рокировки для черных (0 — если не возможна, 1 — если возможна), и, если нужно, одну цифру, характеризующую возможность рокировки для белых (опять же 0 — если не возможна, 1 — если возможна). Построение кода, изображающего позицию, закончено.

Наибольшее число символов содержит код позиции в начальный момент игры: 194. В процессе игры число символов либо остается неизменным, либо уменьшается по мере того, как фигуры покидают доску и играющие лишаются прав рокировки. Наименьшая возможная длина кода позиции равна 72, когда на доске остаются лишь два короля. Условимся в тех случаях, когда код содержит менее чем 194 цифры, дополнять его справа соответствующим числом нулей для того, чтобы его длина всегда равнялась 194.

Мы видим, что любую позицию можно закодировать в виде строки, состоящей из 194 нулей и единиц. Однако не все подобные строки являются изображениями позиций, например строки, являющиеся кодами расположений фигур, при которых короли обеих сторон находятся под шахом или под матом. Следовательно, число шахматных позиций меньше (и значительно), чем число всевозможных строк из нулей и единиц, имеющих длину 194. А число таких строк, как легко подсчитать, равно 2^{194}. Тем самым доказано, что число шахматных позиций конечно потому, что оно не превосходит хоть и большого, но вполне определенного числа.

Читатель может подумать, что учтенная нами дополнительная информация недостаточна для того, чтобы иметь возможность выбрать ход, так как нужно еще знать, не входит ли данное расположение фигур в серию расположений, которая уже повторялась один или два раза. Ответим на это, что при правильной игре, ведущей по кратчайшему пути к выигрышу, повторение серий расположений не может иметь места и потому для нашей цели подобная информация излишня.

Итак, мы убедились, что количество возможных шахматных позиций конечно. Это значит, что оно потенциально перечислимо

(хотя реально их перечислить и невозможно: их так много, что на перечисление ушли бы многие годы).

Выполним теперь (конечно, лишь мысленно) следующий процесс. Отберем все позиции, содержащие мат белым, и отнесем их к классу 0. Далее отберем все позиции, позволяющие при ходе белых сделать мат черным, и отнесем их к классу 1. При белой игре для каждой позиции класса 1 существует хотя бы один ход, превращающий ее в позицию класса 0. Потом отберем все позиции, которые при любом ходе белых превращаются в позиции класса 1, и отнесем их к классу 2. Продолжаем этот процесс, пока он не прекратится из-за того, что просмотрены уже все возможные позиции. При этом, каждый раз формируя класс с нечетным номером, будем включать в него все позиции, каждая из которых допускает только ходы, ведущие к позициям, принадлежащим уже сформированным классам с четными номерами. Отбирая позиции в класс с четным номером, будем включать в него все позиции, каждая из которых допускает хотя бы один ход, ведущий к позиции класса, имеющего номер, на единицу меньший, чем номер формируемого класса. После окончания описанного процесса (обозначим наибольший полученный при этом номер класса через M) останется некоторое количество позиций, не вошедших ни в один из классов (например, позиции, содержащие пат белым и некоторые другие). Эти оставшиеся позиции отнесем к классу $M+1$.

Из самого способа классификации видно, что если перед шахматистом возникла позиция, принадлежащая классу с нечетным номером n ($n \leqslant M$), то при правильной игре он выиграет за n полуходов (если противник тоже будет играть наилучшим образом) или даже быстрее. Наилучшим ходом при этом будет любой ход, в результате которого для противника возникнет ситуация, входящая в класс номер $n-1$ (четный). Если перед шахматистом возникла ситуация с четным номером m ($m \leqslant M$), то при правильной игре противника выигрыш невозможен. Наилучший ход позволяет проиграть за наибольшее число полуходов. Если же возникла позиция из класса с номером $M+1$, то наилучшим будет ход, который ведет к позиции того же класса, т. е. сохраняет ничейную ситуацию. Заметим, что в каждой позиции наилучших ходов может быть и несколько.

Теперь легко представить себе правило наилучшей игры в шахматы. Оно и является искомым алгоритмом точной игры в шахматы. Вот этот алгоритм.

1. Просмотрев классы позиций, начиная с класса 0, найти свою позицию. По номеру класса, к которому она относится, мы заранее узнаем возможный исход партии.

2. Перебрав возможные ходы, найти наилучший (т. е. при позиции класса $M+1$, ведущей к позиции того же класса, а при позиции всякого другого класса — к позиции класса, номер которого на единицу меньше).

Пользуясь указанным правилом, Вы будете играть наилучшим образом.

Мы видим, что само правило игры весьма просто, но число позиций, которые необходимо сначала рассортировать, а затем просмотреть, является числовым гигантом. Но нельзя ли, видоизменив и, может быть, усложнив это правило, сделать его реально осуществимым? Поставленная таким образом проблема относится

к области равносильных преобразований алгоритмов и является уже не шахматной, а математической проблемой.

После того, как получен хоть какой-нибудь алгоритм точной игры в шахматы (а нам удалось получить, но, к сожалению, неэффективный), его преобразования уже не требуют искусства шахматиста. Существует путь, идя которым, можно имеющийся алгоритм преобразовывать в другие алгоритмы, дающие решение той же самой задачи. Этот путь заключается в следующем. Алгоритм точной игры нужно записать на формальном алгоритмическом языке, для которого разработаны правила формальных равносильных преобразований. В настоящее время такие правила разработаны для алгоритмического языка, получившего название языка логических схем (коротко ЯЛС). Этот язык возник в теории программирования как результат развития операторного метода, созданного в 1953 г. выдающимся советским математиком член.-корр. АН СССР А. А. Ляпуновым. На языке логических схем алгоритм точной игры в шахматы можно записать в виде строки специальных символов. Правила равносильных преобразований алгоритмов, заданных на ЯЛС *, имеют вид формул. Каждое применение формулы превращает имеющуюся строку символов в новую строку, также являющуюся записью алгоритма, причем алгоритма, который равносилен исходному. Выполняя одно или несколько преобразований, мы можем получать разные, но равносильные между собой алгоритмы точной игры в шахматы. Некоторые из них могут быть «лучше», а другие «хуже», чем исходный алгоритм. Нашим критерием качества является эффективность. По существу, этот критерий сводится к тому, чтобы, во-первых, необходимый при выполнении алгоритма объем запоминающих устройств не превосходил объема запоминающих устройств используемой ЭВМ и, во-вторых, время, расходуемое на выполнение алгоритма, не превышало допустимого (например, было не больше 3 мин. 45 с., что обеспечивает установленные нормы: 2,5 ч на первые 40 ходов каждой стороне и по одному часу на каждые последующие 16 ходов).

Обозначим через V_0 объем запоминающих устройств машины. Можно считать, что из двух алгоритмов тот лучше, который требует меньшего объема запоминающих устройств, если хотя бы один из них требует объема запоминающих устройств большего, чем V_0. Если же требуемые объемы запоминающих устройств одинаковы или оба они не превосходят V_0, то из двух алгоритмов тот лучше, который требует меньшего времени. Наша задача состоит в том, чтобы выбрать такую цепочку преобразований, после выполнения которой будет получен эффективный алгоритм. Правда, возникает опасение, что описанный выше неэффективный алгоритм никак нельзя преобразовать в эффективный. Но всякая работа, если она выполняется в первый раз, связана с риском; кроме того, переходя к более подходящей по своим операционным возможностям машине, мы всегда получаем дополнительные возможности улучшения алгоритма.

Конечно, рецепта для выбора цепочки преобразований, которая привела бы к получению эффективного алгоритма, мы не знаем. Но

---

* Язык логических схем и правила равносильных преобразований алгоритмов, заданных на нем, описаны в книге Н. А. Криницкого «Равносильные преобразования алгоритмов и программирование», М., «Сов. радио», 1970.

если бы такой рецепт существовал, то не было бы никакой проблемы и эффективный алгоритм игры в шахматы был бы уже получен. Но, коль скоро нет рецепта, значит задача получения эффективного алгоритма относится к числу проблематичных и, без сомнения, является трудной. Возникает вопрос: что произойдет, если будет найден эффективный алгоритм точной игры в шахматы?

Ответ на этот вопрос дать нетрудно, потому что для одной игры, которой многие увлекались, такой алгоритм был найден. Мы имеем в виду игру «такен», известную также как игра в пятнадцать *, изобретенную знаменитым составителем шахматных задач Сэмюэлем Ллойдом. Увлечение ею достигло своего апогея в 1880 г.; за решения отдельных возникающих в ней задач назначались крупные денежные премии (до 1000 долл.). Но вскоре, после появления математической теории этой игры и составления алгоритма наилучшей игры, интерес к игре в пятнадцать упал, чтобы никогда не возродиться. Значит, если будет найден эффективный алгоритм точной игры в шахматы, интерес к шахматам исчезнет? Безусловно! Но человек, который создаст этот алгоритм, войдет в историю человеческой культуры. Могут сказать, что его слава будет славой Герострата **. По нашему мнению, такая оценка неверна. Герострат не много сделал. Он уничтожил сделанное другими. Изобретатель же точного алгоритма игры в шахматы ничего не уничтожит. Он научит всех делать то, чего сейчас не может делать никто: вести точную игру в шахматы.

Представим себе на минуту, что некто овладел эффективным алгоритмом точной игры в шахматы. Если бы алгоритм был настолько хорош, что его можно было выполнять в уме, этот некто мог бы выступить в роли шахматиста и в результате последовательных участий в ряде матчей и турниров завоевать шахматную корону. После этого он мог бы сам себя развенчать. Но как! Это развенчание было бы более блистательным, чем пожизненное сохранение шахматной короны.

Мы видим, что стать создателем точной игры в шахматы очень заманчиво. Но все же, настолько ли заманчиво, что найдутся люди, которые захотят тратить свой досуг и усилия на его разработку? На этот вопрос ответить трудно. Но можно опять сослаться на историю. Теперь уже на историю одной из математических проблем. Имеется в виду так называемая великая теорема Ферма, сформулированная еще в XVII в., и полностью не доказанная (и не опровергнутая) по сей день ***. Многие сотни людей (в своем большинстве не специалисты в области математики) потратили годы своей жизни на попытки доказать эту теорему. Некоторыми из них руководили научные интересы, но многих соблазняла денежная премия (в размере 100 000 марок), назначенная тому, кто первым даст доказательство этой теоремы (заметим, между прочим, что эта премия в конце концов была аннулирована). Если ради ста тысяч марок многие не жалели ни труда, ни времени, то, может быть, найдутся

---

* Игра в пятнадцать описана в книге Я. И. Перельмана «Живая математика», см. например, изд. 10-е, «Наука», 1974.

** Герострат, желая прославиться, в 356 г. до н. э. сжег в Эфесе храм Артемиды, выдающееся произведение античного искусства.

*** **См. Курант Р. и Робинс Г.** Что такое математика, изд. 2-е. **М., «Просвещение», 1967, с. 67.**

и энтузиасты, которые будут готовы потрудиться ради более благородной цели.

Вернемся теперь к основной нашей теме — к теме данной книги. Ботвинник не ставит перед собой цели создать алгоритм точной игры в шахматы. Он хочет переложить на точный язык команд машины свой собственный метод игры в шахматы. На его работу можно взглянуть по крайней мере с двух точек зрения. Одна из них — это точка зрения шахматного мастерства. Ведь Ботвинник — один из крупнейших шахматистов нашего столетия. Пользуясь своим методом, он не раз одерживал победы. Вторая точка зрения позволяет понять трудность задачи, которую поставил перед собой Ботвинник. Эта трудность сравнима с трудностью, которую испытал бы человек, который умеет правильно говорить, но не знает грамматики и приступает к ее составлению (правда, в наша время такие задачи стали легче, чем были когда-то, но все же они очень трудны).

Ботвинник вводит понятие неточной задачи, понимая под этим каждую задачу, для решения которой нужно произвести переработку столь большого количества информации, что при существующих средствах эта переработка реально не может быть выполнена. Задачу определения хода в заданной шахматной позиции он и относит к числу неточных. Анализируя понятие неточной задачи, М. Ботвинник приходит к выводу, что невозможность произвести переработку информации равноценна отсутствию определенной ее части. Исходя из этого, он предлагает своеобразный подход к решению таких задач, названный им методом горизонта. Метод горизонта заключается в том, что учитываемая при решении задачи информация ограничивается (мы видим окружающий нас мир только до горизонта). Однако, если при анализе информации мы переходим от одних ее частей к другим, учитываемая информация становится другой (горизонт перемещается). Характерным при методе горизонта является то, что в состав учитываемой информации включается именно та ее часть, которая наиболее существенна при решении задачи. При каждом конкретном применении метода горизонта определение понятия горизонта производится вполне естественным способом. При решении шахматной проблемы под горизонтом понимают заданное число полуходов.

Тот самый путь, который привел Ботвинника к методу горизонта, привел его к выяснению того, что задачи перспективного экономического планирования также могут быть отнесены к классу неточных задач. В результате этого разрабатываемый Ботвинником алгоритм игры в шахматы приобрел еще один аспект, кроме шахматного. Он стал пробным камнем для разработки методов решения задач перспективного планирования.

Рассматривая класс неточных задач, М. Ботвинник, по существу, затрагивает проблематику новой научной дисциплины, известной под названием теории сложных систем. При изучении многих реальных систем некоторые из них удается описать с помощью алгоритма. Делается это так: внешние воздействия на систему и ее ответы на них изображаются в виде наборов чисел, а действия системы описывают в виде алгоритма, точного правила, позволяющего по данным внешним воздействиям вычислить отвечающую им реакцию системы. Если это удалось сделать, то алгоритм, описывающий действия системы, является математическим описанием самой системы.

Систему, описанную в виде алгоритма, можно автоматизировать, заменить ее комплексом машин, управляемых с помощью ЭВМ.

Однако это возможно только при условии, что полученный алгоритм эффективен, не требует при современном уровне техники слишком большого расхода времени или слишком много места в запоминающих устройствах. Когда речь идет о реальных системах, а не просто о решении задач, то выполнение алгоритма может быть связано также с расходованием каких-нибудь ресурсов, отличных от времени и объема запоминающих устройств, например с расходованием энергии (машинами, управляемыми с помощью нашего алгоритма), материальных средств. Поэтому можно сказать, что сложной системой называется алгоритм, выполнение которого требует большего расхода ресурсов, чем предназначено для этой цели. Основной задачей теории сложных систем является задача построения системы, которая, не являясь сложной, по своим результатам близка к рассматриваемой сложной системе. Легко видеть, что введенные Ботвинником в рассмотрение неточные задачи сродни сложным системам. Метод горизонта является одним из методов решения основной задачи теории сложных систем. Ботвинник рассматривает задачу определения хода при заданной шахматной позиции как сложную систему.

Но такой подход к решению проблемы делает удобным для изложения ее идей применение языка кибернетики. В соответствии с этим первую часть своей книги Ботвинник посвящает рассмотрению структуры систем, которые он называет полными системами управления. Проблему шахматной игры, которая в нашей терминологии является сложной системой, он сводит к несложной системе, достаточно близкой (по его мнению) к ней по своим результатам. При этом шахматы рассматриваются как иерархическая полная система управления, имеющая три уровня. На нижнем уровне этой системы находятся фигуры, каждая из которых действует на некоторой траектории и имеет свою цель игры, следующий уровень образуют группы фигур, взаимодействующие в так называемых зонах игры и преследующие свои цели. Высшим уровнем системы является совокупность всех фигур. Этой совокупности соответствует общая главная цель. Цели, преследуемые на всех трех уровнях, определенным образом согласованы, причем цели более низкого уровня имеют подчиненное значение по отношению к целям более высокого уровня. Для шахмат содержанием целей на всех трех уровнях является стремление выиграть материал. Анализ целей, стоящих перед полной системой управления и ее составными частями, М. Ботвинник вполне основательно считает одной из важнейших проблем при создании алгоритма шахматной игры, что и подчеркивает названием данной книги.

Алгоритм шахматной игры Ботвинника решает задачу о выборе хода, осуществляемом такой трехступенчатой системой управления, причем рассмотрение информации ограничивается с помощью метода горизонта, о котором мы уже говорили. Итак, действительно, алгоритм Ботвинника не является алгоритмом точной игры в шахматы. Теперь попытаемся сравнить метод Ботвинника решения проблемы выбора хода с другими методами, которые были использованы до сих пор.

Как известно, еще в 1949 г. американским математиком Клодом Шенноном были высказаны основные соображения о возможности машинной игры в шахматы. Идея игры в общем сводилась к тому, чтобы на глубину, равную некоторому выбранному числу полуходов, просмотреть все возможные свои ходы, на каждый из них — все возможные ответы противника и т. д. При этом используется некоторая

функция, присваивающая каждой возникающей позиции некоторую цену. Такая функция называется оценочной. В качестве наилучшего выбирается такой ход, который после конца перебора позволяет прийти к позиции, имеющей наибольшую цену, при условии, что противник каждый раз так отвечает на наши ходы, что оценочная функция при этом к концу перебора получает, поскольку это от него зависит, как можно меньшее значение. Легко видеть, что при таком алгоритме охватываются все фигуры, стоящие на доске, и анализу подвергаются все ходы, как представляющие собой интерес, так и заведомо бесполезные. Шеннон отмечал, что полный перебор всех возможных ходов требует чрезмерного расхода машинного времени и высказал пожелание, чтобы будущие разработчики шахматных программ применяли какие-нибудь методы, суживающие область перебора путем исключения из рассмотрения заведомо бессмысленных ходов.

Таким образом, Шеннон предложил строить шахматные программы на следующих трех китах: 1) перебор ходов на заданную глубину; 2) оценочная функция позиций; 3) ограничение области перебора. Все известные до настоящего времени шахматные программы покоятся на этих китах. Самым капризным из этих китов оказался третий. До сих пор нет ни одного метода ограничения области перебора, который бы вместе с бессмысленными ходами не исключал из рассмотрения и самые ценные. Второй кит (оценочная функция) тоже подводит. Шеннон рекомендовал саму идею оценочной функции. Конкретную оценочную функцию он предложил всего лишь в качестве наводящего примера. Но до сего дня ничего лучшего, чем эта, приведенная в качестве примера функция, не придумано. А эта функция определяет цену позиции в зависимости от ряда факторов, каждый из которых в большинстве случаев (как это следует из учебников шахматной игры) способствует успеху, но далеко не всегда! Такая оценочная функция не работает в эндшпиле и, как утверждают шахматисты, не подтверждается анализом партий мастеров шахматной игры. Правда, сама по себе идея оценочной функции, без сомнения, правомерна. Если читатель согласен с тем, что среди возможных позиций есть худшие и лучшие, то он тем самым признает существование оценочной функции, которая имеет тем большее значение, чем лучше оцениваемая позиция. Но, наверное, оценить позицию не так-то просто. Что же касается разработчиков шахматных программ, то им мы пожелаем побольше фантазии и посоветуем увеличить творческие усилия.

Отметим, что и у первого кита есть ахиллесова пята. Перебор дает тем лучшие результаты, чем больше его глубина. Но с увеличением глубины сильно возрастает время перебора. Тем не менее этот кит надежен.

В чем же отличие метода Ботвинника от метода, предложенного Шенноном? Прежде всего в том, что у Ботвинника нет статической функции, оценивающей позиции. В его концепции основу составляют не позиции, а траектории. Каждая траектория «пронизывает» большое число позиций. Искомый ход выбирается по одной из траекторий. Руководством для выбора хода является цель игры. Перебор возможных ходов имеется и у Ботвинника, но каждый ход является не столько шагом, ведущим от одной позиции к другой, сколько движением по траектории, которое наиболее способствует достижению цели игры.

Некоторые математики, знакомившиеся с методом Ботвинника, считают этот метод лишь новой реализацией идей Шеннона. Выбор траекторий они оценивают как удачный способ отбрасывания бессмысленных ходов, т. е. как способ ограничения области перебора. Анализ действий на траекториях они хотят рассматривать как перебор ходов, имеющих смысл. Выбор хода, наилучшим образом соответствующего цели игры, они представляют как выбор с помощью функции, определяющей цену позиции, но заданной в замаскированном виде. Отчасти это так, но все же против такой «прокрустации» нужно категорически возражать. Если основными элементами структуры метода Шеннона являются позиции, которые можно условно представить себе в виде различных между собой поверхностей, то основными элементами структуры метода Ботвинника являются траектории и их пучки, которые можно представлять себе линиями, ортогональными поверхностям — позициям.

Сказанное не означает, что после соответствующих исследований нельзя построить алгоритм шахматной игры, основанный на методе Шеннона и эквивалентный алгоритму Ботвинника. Но я не вижу реально осуществимого способа для выполнения такой работы и в то же время отчетливо вижу, что алгоритм Ботвинника не является алгоритмом, основанным на методе Шеннона.

Как уже было сказано, разработка Ботвинником алгоритма и, в частности, программы игры в шахматы близится к концу. Как же оценивает сам Ботвинник тот эффект, который произведет в шахматном мире появление программы, играющей в силу гроссмейстера (допустим, что этого удастся достигнуть)? Он считает, что это не уменьшит интерес шахматистов к шахматной игре. И он, безусловно, прав. Алгоритм Ботвинника, не являясь алгоритмом точной игры, не приведет к превращению шахматной игры в механическое выполнение вычислительных процедур. Кроме того, не являясь алгоритмом точной игры, он не может быть и алгоритмом наилучшей игры. Перед шахматистами останется вполне реальная цель: играть лучше машины, играть лучше Ботвинника. А значит, шахматная игра не потеряет своего спортивного характера.

Хорошо, скажет читатель, предположим, что алгоритм Ботвинника уже завершен и что программа шахматной игры уже готова. Что же мы имеем? Законный вопрос.

Такая программа будет точным математическим описанием метода игры в шахматы, разработанного Ботвинником, т. е. она будет законченным и точно изложенным произведением шахматной науки, произведением, которое можно будет сколько угодно раз «читать», которое, пожалуй, позволит всегда получить ответ на вопрос: как бы сыграл Ботвинник в такой-то позиции?. В отличие от книг и учебников, посвященных вопросам шахматной игры, которые всегда содержат немало неясностей, программа игры в шахматы будет совершенно точным документом. Но такой документ не должен содержать ошибок. Именно поэтому очень важным и очень нужным этапом разработки программы шахматной игры будет проведение экспериментальных игр и устранение недосмотров, допущенных при составлении программы.

Пожелаем Ботвиннику и его соратникам успешно завершить начатую ими и уже далеко продвинувшуюся большую работу.

Н. А. Криницкий.

От автора

Задача данной книги — ознакомить читателя с алгоритмом игры в шахматы, основанным на выигрыше материала и контроле полей, а также реализацией этого сложного алгоритма в виде машинной программы.

В 1968 г., когда в издательстве «Наука» вышла книга «Алгоритм игры в шахматы», автора поддержал лишь один видный специалист — Н. А. Криницкий. Сейчас положение изменилось: повлияли на перемены как слабость достигнутых результатов всех тех работ, которые основывались на оценочной функции (цели игры), предложенной К. Шенноном, так и первые положительные результаты, полученные при реализации нового алгоритма. До завершения работы еще далеко, но то, что сделано, обнадеживает.

Автор хочет выразить искреннюю благодарность тем, кто содействовал изданию этой книги: Н. А. Криницкому, Д. Б. Юдину, Э. М. Вайсборду, Е. С. Геллеру, а также своим товарищам по работе: Д. Н. Лозинскому, Л. М. Полтавец, Б. М. Штильману, А. Д. Юдину и Л. В. Панфиловой (за оформление рукописи).

ВВЕДЕНИЕ

Многое из изложенного в этой книге всего лишь гипотеза, поэтому читатель вправе отнестись к работе критически. Конечно, если бы алгоритм игры в шахматы был уже представлен в виде машинной программы, которая играла бы с большой силой, для скепсиса было бы меньше оснований. Однако по разным причинам работа над программой откладывалась и началась лишь в августе 1972 г.

Позволительно задать вопрос: а не следовало ли повременить с публикацией этого труда, не подождать ли, пока работа над программой даст результаты? Вряд ли это было бы лучшим решением, так как многие специалисты, работающие над созданием алгоритмов, программ (в том числе и шахматных), вероятно, найдут для себя кое-что полезное, прочитав эту книгу. Уже в этом случае ее издание оправдано.

Сначала был сделан алгоритм шахматной игры, затем появились некоторые обобщения — они напрашивались. В книге все изложено наоборот: сначала обобщения, а затем алгоритм, который является как бы иллюстрацией приложения общих методов.

ПОЛНАЯ И ЧАСТИЧНАЯ СИСТЕМЫ УПРАВЛЕНИЯ

Полной системой управления будем называть такую систему, которая выполняет три функции: 1) получение информации, 2) ее переработку (принятие решения) и 3) исполнение решения (рис. 1). При этом вторая функция связана с программой, содержащей подпрограмму самообучения. Все эти функции кибернетические.

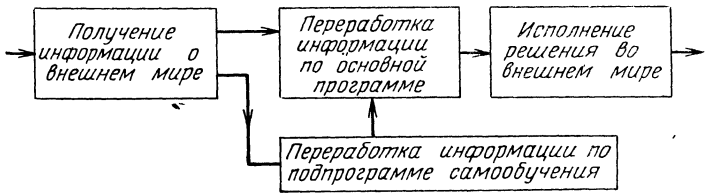

Рис. 1. Схема полной системы управления.

Есть еще и некибернетические функции, например энергоснабжения (она непременно должна быть) и воспроизводства, или размножения. Однако к проблемам управления, как таковым, эти функции отношения не имеют, и поэтому в дальнейшем рассматриваться не будут.

Первая кибернетическая функция ясна. Без получения информации о внешнем мире нечего перерабатывать.

Третья функция также ясна. Решение должно быть исполнено, что непременно связано с внешним миром.

Вторая функция более сложная. Сначала перерабатывается первичная информация. В процессе переработки она может разрастаться, в некоторых случаях практически беспредельно *. Информация перерабатывается по фиксированной (на данный момент времени) программе. Но, как уже отмечено, эта программа (основная) может меняться в соответствии с фиксированной подпрограммой самообучения.

Система управления может быть неполной (частичной), если отсутствует, например, подпрограмма самообучения.

ЕСТЕСТВЕННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Приведем классификацию естественных систем управления, которая, впрочем, не обязательна (табл. 1). Автор все же надеется, что специалисты отнесутся к ней благосклонно.

Таблица 1

Классификация естественных систем управления

Естественная система управления, относящаяся к	Функции системы управления
Неживой природе	1-я и 3-я, отсутствует 2-я
Растительному миру	1-я, 2-я, 3-я, но во 2-й функции отсутствует подпрограмма самообучения
Животному миру	1-я, 2-я, 3-я (полная система управления)

Таблица, несомненно, требует пояснений. Почему отсутствие второй функции у системы управления подразумевает систему, которая характерна для неживой природы? Неживая природа мертва с кибернетической точки зрения, если принять, что главное в системе управления — переработка информации. Получение информации от внешнего мира сводится к внешнему воздействию,

* В этих случаях может потребоваться такой объем запоминающих устройств, каким устройство по переработке информации не располагает.

а исполнение решения — к реакции на это воздействие, происходящей по законам неживой природы.

Что бы ни делалось на белом свете, земля притягивает к себе все, что находится на ее поверхности. То, что неживая природа исполняет одни и те же решения, отнюдь не значит, что в «мертвом» мире нет никаких изменений. Изменения, несомненно, есть, но они происходят не потому, что меняются исполнения решений, а потому, что меняются результаты действий множества исполнительных органов, но каждый исполнительный орган (пока он не разрушен в процессе взаимодействия с другими) консервативен в своих действиях.

Такая частичная система управления предельно консервативна, и первый закон Ньютона — очевидное подтверждение этому. Изменения в неживой природе происходят не от отсутствия консерватизма, а от случайного взаимодействия частичных консервативных систем управления.

Второй тип (также частичной системы управления) — сочетание получения информации, переработки ее без самообучения и исполнения решения. Если определенной информации соответствует определенное исполняемое решение, то перерабатывать по сути дела нечего: это все равно, что справочная система. Видимо, так управляется растительный мир. Исполняемое решение зависит от полученной информации, однако информация столь примитивна, что переработка ее весьма проста: солнце взошло — подсолнух к нему повернулся...

И здесь частичная система управления достаточно консервативна, но не исполнением решения, а соответствием решения полученной о внешнем мире информации. Таким образом, исполняемое решение не одно и то же, оно различно, но однозначно зависит от первичной информации. Здесь решение принимается по справочному методу; консервативен в данном случае сам метод в той конкретной форме, в которой он действует. Эта форма, конечно, меняется хаотически, по случайным причинам (вряд ли есть смысл пересказывать здесь теорию Дарвина).

Наконец, полные системы управления, которые соответствуют животному миру. Объем первичной информации возрастает необычайно. Видимо, это связано со способностью системы управления к самостоятельному передвижению — связь с внешним миром становится

многогранной. Соответствие исполняемого решения полученной информации без ее сложной переработки невозможно. Переработка информации только по справочному методу недостаточна, и она происходит по определенной программе. Эта программа в принципе не консервативна: она способна меняться в соответствии с подпрограммой самообучения во время действия (жизни) системы. Ну, а подпрограмма самообучения передается по наследству в некотором начальном виде. Конечно, и она меняется в процессе жизни.

Таким образом, при первом типе системы управления, когда нет устройства по переработке информации, консервативно исполняемое решение. При втором типе, когда есть и датчики, и примитивная переработка информации, и исполнительный орган, исполнение решения многообразно, но однозначно (по справочному методу) соответствует получаемой информации о внешнем мире (консервативен лишь метод). А при третьем типе разнообразно исполняемое решение, изменчива программа переработки информации, консервативна лишь начальная информация о способе изменения этой программы.

Не следует забывать, что все эти консервативные элементы системы управления меняются по случайным причинам. Укажем еще, что полная система управления может содержать дополнительно программу переработки информации и по справочному методу, аналогичную той, которая характерна для растительного мира.

ИСКУССТВЕННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Некоторые полные системы управления настолько совершенны, что они могут сознавать свое несовершенство. Наиболее совершенными из всех естественных полных систем управления являются люди. Поэтому люди с далеких времен пытаются искусственно улучшить элементы своих систем или, точнее говоря, дополнить или заменить естественные элементы своих систем искусственными.

Человек, прежде всего, страдал от недостаточной мощи своего исполнительного органа — мышечной силы, ибо на земле обитали другие живые существа, обладавшие большей физической силой, хотя и более слабым

устройством переработки информации. Приручение диких животных не могло решить эту проблему радикально: требовался искусственный исполнительный орган. И еще в древние времена человек добился первого большого успеха, успеха принципиального, когда научился использовать энергетические ресурсы, скрытые в окружающей природе (речь идет об открытии огня). Открытие атомной энергии, хотя и привело к использованию неизмеримо бо́льших энергетических возможностей, ничего с кибернетической точки зрения принципиально нового не дало.

Итак, люди создали серию искусственных исполнительных органов системы управления. Иногда они используются вместо естественного элемента, иногда дополняют его — это зависит от конкретных задач, которые система управления должна исполнять.

Использование дополнительных естественных датчиков информации не могло радикально помочь делу, и поэтому люди стали искать искусственные датчики. Но если в создании искусственного исполнительного органа люди одержали принципиальную победу очень давно, то решающий успех в создании искусственного устройства по передаче информации достигнут совсем недавно (речь идет об открытии радио). Первая радиограмма А. С. Попова была лишь первой ласточкой, началом новой эпохи, когда возможности управления расширились фантастически.

Сейчас искусственные элементы получения информации и исполнения решения во многих важных случаях дополняют естественные элементы, входящие в систему управления, которая называется «человек», а иногда и заменяют их. Но один естественный элемент такой системы управления пока является монополией человека— устройство по переработке информации (мозг).

Система управления должна быть гармоничной, если так можно выразиться. Мощный исполнительный орган должен исполнять разумное решение — ошибка может обойтись очень дорого, она будет «мощной» (а разумное решение должно быть, в свою очередь, обеспечено необходимой информацией). За многовековую историю в области получения информации и исполнения решения люди добивались скачкообразных успехов, а мозг человека развивался эволюционным и, несомненно, более скром-

ным путем. Отставание в деле создания искусственного интеллекта, несомненно, опасно.

Таким образом, чтобы иметь сбалансированную мощную систему управления с искусственными элементами надо создать еще один искусственный элемент системы управления, а именно искусственный интеллект, который может как дополнять человеческий мозг, так и в ряде случаев заменять его.

Искусственные системы могут содержать искусственные элементы в нужных комбинациях, тогда как естественные — лишь в определенных комбинациях, соответствующих табл. 1. Искусственные элементы могут существовать независимо от систем управления и допускать подключение к ним, тогда как отдельное существование естественных элементов исключено*; это же относится и к интеллекту (устройству по переработке информации, работающему по определенной программе). Искусственная полная система управления может иметь сложную программу переработки информации и не содержать подпрограммы самообучения.

МНОГОСТУПЕНЧАТЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Отдельные системы управления могут образовывать новую систему управления, являющуюся их совокупностью. Рассмотрим один тип такой совокупности, который, по-видимому, весьма распространен.

Примем, что отдельные системы управления образуют совокупность систем управления, которая является одной из систем, образующих новую, еще более сложную совокупность. Причем решения, принятые совокупностью систем, обязательны для систем, входящих в ее состав (эти решения учитываются при переработке информации в отдельных системах).

Для иллюстрации приведем пример управления заводом. Каждый рабочий представляет собой, с кибернетической точки зрения, полную систему управления. Рабочие образуют совокупность — бригаду, бригады — цех, цеха — завод. Мы имеем многоступенчатую систему управления, в которой нижестоящая ступень при перера-

* В настоящее время ведутся работы, в которых исследуется возможность использования живых органов в искусственных системах.— *Прим. ред.*

ботке информации использует информацию о решении вышестоящей.

Можно заметить, что это неточно, что при некоторых экономических системах управления решения нижестоящих систем влияют на решение вышестоящих или, как говорят, контролируют их. Да, влияют и контролируют в соответствии с программой по переработке информации данной совокупности элементов. Но когда решение совокупности принято, оно обязательно для нижестоящих систем в соответствии с их программой по переработке информации.

Итак, многоступенчатая система управления заслуживает самого пристального внимания с точки зрения не только теории, но и практики.

В простейшем случае мы имеем двухступенчатую систему, когда системы образуют одну совокупность. Конечно, самым радикальным было бы изучение n-ступенчатой системы, но это задача не из легких. Ограничимся рассмотрением двух (или трех) ступеней. Видимо, в дальнейшем это будет хорошей основой для решения задачи в целом.

Здесь нельзя не признать удачной идею К. Шеннона, который предложил математикам работать над созданием машинной программы шахматной игры — модель для исследования хорошо выбрана. Игру в шахматы следует считать, как минимум, двухступенчатой системой управления. Каждая фигура определенного цвета (вместе с шахматистом) образует простейшую систему управления. Все фигуры одного цвета (вместе с шахматистом) образуют совокупность, являющуюся более сложной системой. Поэтому общие проблемы, которые возникают при изучении этой полной системы управления (игры в шахматы), вероятно, типичны и для других двухступенчатых (а может быть и многоступенчатых) систем управления.

Конечно, если шахматные фигуры одного цвета — элементы совокупности фигур того же цвета, совокупности, которая имеет свою программу для переработки информации, то может показаться, что фигура (элемент) лишена подобной программы и поэтому аналогия с реальной двухступенчатой системой управления сомнительна. В действительности дело обстоит (или, точнее, может обстоять) иначе: все зависит от качества программы. У шахматиста (естественного или искусственного) одно

устройство по переработке информации. Но программа у этого шахматиста может быть такой, что это одно устройство обслуживает и элементы и всю совокупность их (нечто подобное происходит в деловом мире, когда мелкие фирмы не имеют своих бухгалтерий, а доверяют вести бухгалтерские операции крупной счетной фирме). Во всяком случае, в моем алгоритме это предусмотрено, а в программах, сделанных по Шеннону, видимо, отсутствует.

Получить первичную информацию о шахматной позиции, правилах игры — дело несложное, исполнить решение (сделать ход) — и того проще. Главная трудность при создании шахматного автомата — создать программу по переработке информации. Поэтому предложение Шеннона по сути дела состоит в создании искусственного интеллекта (шахматиста).

О ЦЕЛИ ИГРЫ

Цель игры, цель работы системы управления является основным фактором, определяющим всю проблему. Исполнение решения должно соответствовать поставленной цели. Цель игры — это эталон, по которому мы оцениваем работу системы управления.

Может показаться, что цель игры определяется элементарно; это было бы заблуждением. Точная цель игры (например, мат) может оказаться ошибочной, дезорганизующей работу системы. Наоборот, паллиативная (неточная) цель может быть правильной, оптимальной для данной системы. В чем же здесь дело?

Все зависит от возможностей системы управления. Если поставленная цель такова, что получить соответствующую информацию нельзя, или устройство по переработке информации не в силах переработать всю необходимую информацию, или программа по переработке информации не соответствует поставленной цели, или, наконец, исполнительный орган не приспособлен для исполнения необходимого решения, система управления оказывается неработоспособной и ничто не сделает ее работоспособной.

Наоборот, если цель игры отличается от точной, но удовлетворительна и эта паллиативная цель гармонирует

с системой управления, то система оказывается работоспособной, а результаты ее работы эффективными.

Итак, точная цель игры может привести к неудовлетворительным итогам, а паллиативная цель — дать неплохой результат. Это относится к действующим системам управления и тем более к проектируемым. Особенно внимательным следует быть при составлении программы переработки информации. Здесь удачный выбор цели игры имеет решающее значение.

Мы рассматриваем (как минимум) двухступенчатую систему управления, состоящую из элементарных систем. Цель игры должна быть и у элементов, и у их совокупности. Видимо, у элементов и их совокупности цели должны быть однотипны, но конкретная их форма должна быть дифференцированной, различной. Трудно представить себе работу двухступенчатой системы управления, если у элементов один тип цели, а у их совокупности — другой. Вряд ли может успешно работать и такая система, если у элементов формально та же цель, что и у их совокупности. В этом случае теряется конкретность, дифференцированность, индивидуальность цели элементов, что (как будет видно из дальнейшего) может неблагоприятно отразиться на работоспособности системы. При этом необходимо учитывать, что неточная (см. с. 34) задача, по сути дела, является задачей перспективного планирования. Принятие решения в таких задачах должно исходить как из текущих потребностей, так и из той ситуации, которая может возникнуть в будущем. Поэтому цель игры элементарных систем и их совокупности единая по типу, видимо, имеет две составляющие: первую, учитывающую текущий момент, и вторую, учитывающую перспективу.

Далее будет показано, что в шахматах паллиативная типовая цель (выигрыш материала) делится у совокупности систем на две подцели: текущую (выигрыш материала в данный момент) и перспективную, связанную с контролем тех полей, где происходит шахматное сражение, для того, чтобы в будущем эти благоприятные условия (контроль полей) использовать для выигрыша того же материала.

Выше были оценены заслуги Шеннона в постановке задачи о создании искусственного шахматиста. Но, по мнению автора, необходимо отметить и его ошибку в определении цели шахматной игры, цели той проекти-

руемой системы управления, которую можно назвать «искусственный шахматист».

Известно, что по Шеннону цель шахматной игры (что, в общем, и определяет оценку позиции и выбор хода) — это максимизация линейной функции подвижности фигур, владения центром доски, пешечной структуры, безопасности позиции короля, материального соотношения сил сторон и т. д. Цель эта, подытоживающая многовековой опыт шахматистов, справедлива в том смысле, что каждая из составляющих этой цели справедлива с точки зрения оценки большинства шахматных позиций, которые встречались в шахматной практике. Однако составляющие этой цели, справедливые для большинства этих позиций, оказываются (или могут оказаться) фальшивыми (особенно в своей совокупности), если оценивать какую-либо конкретную позицию.

Уже приходилось отмечать, что такая цель игры неизбежно приводит к шаблонной, общей оценке позиции. Но это не основной дефект работы шахматного автомата по Шеннону. Главный недостаток, видимо, заключается в том, что элементы (фигуры одного цвета) не имеют своих индивидуальных целей, их цели обезличены и полностью совпадают с целью игры совокупности элементов (фигур одного цвета). Отсутствие индивидуальной конкретной цели у фигуры приводит к неизвестности: в каком направлении в своих корыстных целях фигура должна двигаться; в процессе перебора она должна перемещаться во всех направлениях, ее движение приобретает хаотический характер. Перебор всех ходов фигур становится неизбежным, и, хотя сам Шеннон указал на необходимость исключения ходов из перебора, не имеющих смысла, ни в одной известной шахматной программе для ЭВМ эта вторая задача не решена удовлетворительно. Видимо, это происходит по весьма уважительной причине: при цели игры, связанной со всей доской, эту задачу и решить нельзя.

ОПАСЕН ЛИ ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ?

Это весьма серьезный вопрос. Искусственный интеллект должен быть сильным, он настолько должен в перспективе превосходить мозг человека, насколько, скажем, термоядерная бомба превосходит артиллерий-

ский снаряд. Не выйдет ли тогда искусственный интеллект из-под контроля человека, не окажется ли в будущем человек на земле в таком же положении, в каком сейчас находится комар?

Человек силен своим сознанием. Опасность, очевидно, возникнет тогда, когда «сознание» искусственного интеллекта будет мощным. Поэтому попытаемся сначала определить, что такое сознание (с кибернетической точки зрения), чтобы при проектировании искусственного интеллекта не совершить роковой ошибки.

Представим, что искусственная полная система управления работает без подпрограммы самообучения, т. е. по фиксированной (консервативной) программе переработки информации (принятие решения), программе, приспособленной для решения какой-либо узкой задачи. Никакой опасности в этом случае быть не может. Пока искусственный интеллект пребывает в этом состоянии, как бы он не был силен, он совершает одну и ту же полезную работу. Это важный факт. В этом случае сознание у искусственного интеллекта отсутствует, мы имеем дело со способным интеллектуальным роботом, но этот робот не осознает ни внешний мир, ни самого себя. Человечество в безопасности.

Но оптимальна ли эта фиксированная программа по переработке информации у такого робота? На это мало надежд, если не ввести обратную связь от внешнего мира, т. е. подпрограмму самообучения, в основе которой лежит сравнение поставленной цели x достигнутым результатом. Как только появляется эта обратная связь, как только система управления начинает получать информацию от внешнего мира о результатах своей работы и эта информация воздействует на программу переработки информации, система управления начинает «чувствовать» внешний мир и самое себя — появляются зачатки «сознания».

Если программа обслуживает узкую задачу и эта программа единственная у системы управления, то сознание в «зачаточном» состоянии. Если же ввести в устройство по переработке информации превеликое множество программ при наличии подпрограммы самообучения, можно считать, что у этой системы управления аналогично человеку присутствует сознание.

Поэтому напрашивается путь, который может привести к созданию сильной программы для мощной системы

управления (с помощью подпрограммы самообучения), но безопасный для человечества: эту подпрограмму включать избирательно для той конкретной программы переработки информации, которая нуждается в совершенствовании. Так электронный бык автоматически превращается в электронного вола.

С этой точки зрения напрашивается сравнение сознательной и подсознательной сторон мышления человека. После того, как человек отработает программу решения какой-либо узкой задачи, решение этой задачи идет по «подсознательному» каналу, сознание уже не нужно, подпрограмма самообучения выключена, ибо основная программа отработана. Видимо, это выгодно: подсознательное мышление человека обладает существенно бо́льшим быстродействием, ибо оно минует подпрограмму самообучения.

Начинающий водитель автомашины знает последовательность операций, необходимых для переключения скоростей, но выполняет операции медленно, неуверенно, ибо мыслит еще с использованием подпрограммы самообучения: сравнивает поставленную цель с достигнутым результатом. Спустя некоторое время достигается автоматизм переключения скоростей. Это означает, что основная программа по переработке информации отработана (стала консервативной), подпрограмма самообучения выключена (водитель не сознает своих действий!), ноги и руки водителя при переключении скоростей работают быстро (автоматически), — ему можно выдавать удостоверение на право вождения автомобилем.

Таким образом, сознательная и подсознательная стороны мышления человека с кибернетической точки зрения однотипны и, видимо, и то и другое мышление протекают по однотипной программе. Отличаются они по быстродействию и консерватизму основной программы. Подсознательное мышление протекает быстро, но по консервативной программе, сознательное — медленно и по устанавливающейся программе. Считать подсознательное мышление каким-то недоступным для изучения, имеющим принципиальные отличия от сознательного (как это утверждают некоторые ученые), видимо, нет оснований. Если человек не осознает своих действий, это не значит, что действия непознаваемы. Сильный искусственный интеллект, который будет создан и не будет иметь сознания (подпрограммы самообучения), будет иметь фиксиро-

ванную консервативную программу по переработке информации, т. е. подсознательное мышление. Однако его мышление вполне «познаваемо», ибо программа задана.

Итак, наличие подпрограммы самообучения неразрывно связано с сознанием.

ОБЩИЕ ТРЕБОВАНИЯ К АЛГОРИТМУ

При разработке программы следует учитывать, что во время ее выполнения не должны быть превышены допустимый объем памяти ЭВМ и допустимое время решения задачи. Поэтому к алгоритму можно, по-видимому, предъявить четыре основных требования.

1. Выделение информации первостепенной важности. Об этом уже было сказано выше, что соответствует рекомендациям К. Шеннона, сделанным еще в 1949 г. Если программе не дано отличать существенное от несущественного, если, например, из всей массы имеющихся на данный момент возможных ходов шахматная программа ЭВМ не может выделить ходы первостепенной важности или, иначе говоря, ходы, имеющие смысл, то остается только один неэкономичный путь — перебор всех ходов.

Мне приходилось слышать мнение некоторых специалистов по этому вопросу, что, де мол, ничего плохого в этом нет. Правда, признают они, человек мыслит, выделяя информацию первостепенной важности, но означает ли это, что ЭВМ должна слепо подражать человеку? Человек — это человек, а ЭВМ — это ЭВМ, и она может перерабатывать информацию по-своему. Автор готов был бы поверить в искренность этих рассуждений, если бы человек перебирал всю информацию, а эти специалисты в своих программах для ЭВМ нашли бы хитрый способ выделения важной информации. Но так как дело обстоит наоборот и в этих программах используется примитивный метод, сокращающий лишь длину вариантов в дереве полного перебора, то теорию об особом методе мышления ЭВМ следует рассматривать как попытку оклеветать ни в чем неповинную ЭВМ, которая, увы, себя защитить пока не может. Рекомендация Шеннона на сей счет представляется безукоризненной.

Добавим, что способность программы выделять информацию первостепенной важности, вероятно, находит-

ся в прямой зависимости от правильно выбранной цели игры.

2. Ограничение длины вариантов перебора. Против этого требования уже никто не может возражать. ЭВМ, как и любое другое устройство по переработке информации, в том числе и мозг человека, имеет ограниченные быстродействие и объем памяти. Между тем решение должно быть принято в ограниченный срок и с имеющейся ограниченной памятью ЭВМ. Единственный способ принять решение в этих условиях состоит в ограничении длины вариантов перебора. Этот метод применяется во всех играющих шахматных программах.

Здесь еще раз отметим важность первого требования о выделении информации первостепенной важности. Есть надежда, что перебор информации, лишь имеющей смысл, приведет к экономии и времени и памяти, т. е. ограничение длины вариантов перебора станет слабее. Конечно, это будет лишь в случае, если отбор важной информации не потребует существенного расхода времени и памяти.

Надо полагать, что и здесь успех зависит от выбранной цели игры.

3. Выделение и экономное исполнение повторяющихся операций. В каждой задаче кроме ситуаций оригинальных неизбежно встречаются ситуации повторяющиеся. Например, в шахматах после того, как четыре с половиной столетия назад окончательно сложились правила игры, повторяющейся операцией является передвижение данной фигуры с одного поля на другое поле свободной от фигур доски. Из партии в партию эта стандартная операция в течение веков переходит в одном и том же застывшем состоянии. Ясно, что эта стандартная операция должна выделяться и исполнение ее программой должно происходить наиболее экономно.

Более или менее очевидно, что программа должна выделять эти повторяющиеся стандартные операции и исполнять их простейшим и скорейшим образом. Это будет способствовать экономному расходованию и времени и памяти ЭВМ, т. е. в конечном итоге более глубокому решению задачи. И здесь успех находится в прямой связи с выбранной целью игры. Если цель игры не обеспечивает выделения стандартных операций и их простейшего исполнения, значит цель выбрана неудачно.

4. Не пересчитывать уже сосчитанного (если достаточен объем памяти) ... Действительно, пусть специалист пришел к выводу, что только сложный, хитроумный алгоритм обеспечит хорошее решение задачи; пусть также до этого действовали сравнительно простые программы, с помощью которых достигались, правда, слабые результаты, но все же какие-то результаты. Естественно, новый сложный алгоритм встречается в штыки и со всех сторон неизбежны возражения: алгоритм сложен, программа будет работать медленно и никакой памяти не хватит, ибо получение решения потребует большого объема работы ЭВМ.

Между тем, это требование в ряде случаев позволяет избежать указанных недостатков. Программа должна отделять информацию, которая меняется, от информации, которая осталась неизменной. Если программа эту измененную информацию способна выделять, то опасаться сложной программы нечего.

Есть надежда, что при этой программе потребуются время и объем памяти, соизмеримые с простой программой (когда неизменяемая информация не выделяется), а решение будет более точным. Однако эту рекомендацию нельзя принимать безоговорочно. Если информация занимает большой объем памяти, а восстановление этой информации — дело простое, то может оказаться выгодным не запоминать, а возобновлять информацию, это зависит от конкретных обстоятельств.

Таковы, на взгляд автора, четыре основных требования к алгоритму.

Посмотрим теперь, как обеспечивает реализацию этих требований цель шахматной игры по Шеннону.

1. В США 1970—1973 гг. Ассоциацией вычислительной техники были проведены чемпионаты шахматных программ для ЭВМ. Профессор М. Ньюборн (под его руководством проходил первый чемпионат 1970 г. в Нью-Йорке) в своем обзоре первых двух чемпионатов указывал, что все программы, участницы турнира, были сделаны по Шеннону (они отличались лишь по языку программирования, типу ЭВМ и т. п.). Таково свидетельство очевидца.

Если за 22 года не удалось избавиться в шахматных программах от «полного» перебора ходов, невыгоды которого еще в 1949 г. были отмечены самим К. Шенноном, значит есть какая-то очень уважительная причина, пре-

пятствующая продвижению вперед. И эта причина (по мнению автора) — неудачная цель игры, которая связана со всей доской, с массивом 8×8. Ни одно поле этого массива не выделяется поставленной целью, поэтому элементарные системы управления (фигуры) не имеют своих конкретных, индивидуальных целей. Цель, связанная со всем массивом 8×8, не дает никаких рекомендаций элементарной системе в отношении направления движения. Между тем фигура только и умеет, на первый взгляд, что двигаться, а раз рекомендаций на сей счет нет, значит надо перебирать ее движения во всех направлениях, т. е. необходим полный перебор возможных ее передвижений.

Мы видим, что ошибочная цель игры, не обеспечивающая конкретности цели и (в этом понимании) осмысленности действий элементарных систем, неизбежно приводит к слабым результатам.

Добавим, что если и были сделаны попытки избавиться от перебора всех ходов (это стало ясно после первого шахматного чемпионата мира среди ЭВМ в Стокгольме в августе 1974 г.), то они не дали лучших результатов, ибо цель игры не содействовала успешному выделению ходов, имеющих смысл.

2. Ограничение длины вариантов перебора реализуется примитивно. Ограничения по сути дела нет, а просто-напросто, когда время счета истекает, перебор прерывается и производится оценка его вариантов. Программа следит при этом, чтобы варианты были равной длины и в этом есть некоторая маскировка: ограничивается не время перебора, а длина перебираемых вариантов в полуходах (например, в программе-победительнице чемпионатов США длина перебираемых вариантов, как правило, равна четырем полуходам). Однако это ограничение (подравнивание длины вариантов) в основном связано с ограничением времени перебора: в пределах этого времени надо считать все, а затем во всех вариантах (кроме некоторых) счет прекращать.

Представим себе садовника, который, подравнивая ветки деревьев по заданному шаблону, формально исполнял бы инструкцию и удлинял бы ветки, привязывая к ним прутики, чтобы все они были равной длины. Вряд ли эта бессмысленная работа была бы встречена с одобрением. Многие программисты заставляют ЭВМ так работать и утверждают при этом, что все это разумно.

Итак, и второе требование к алгоритму (при полном переборе вариантов), видимо, выполняется неудовлетворительно.

3. С выделением и исполнением повторяющихся стандартных операций дело обстоит не лучше. Когда неизвестно, в каком направлении фигура должна передвигаться, и приходится перебирать все возможные передвижения, то стандартной операцией становится перемещение фигуры за один полуход. Стандартная операция становится очень простой, в то время как стандартизировать надо предельно сложные операции; именно в этом случае и будет получена наибольшая выгода. Стандартизация предельно простых операций существенных выгод дать не может.

Поэтому при определении цели игры по Шеннону, цели, связанной со всем массивом 8×8, по существу, не удается выполнить и этого требования.

Посмотрим, как же обстоит дело с последним, четвертым требованием.

4. Здесь автор должен быть осторожным, так как не знает тонкостей действующих программ. Может быть, удается избежать повторных расчетов, может быть, нет. Думаю, что не удается, но уверенности на этот счет у меня нет. Да это и несущественно. Уже ясно, что, когда цель игры связана со всей доской, с массивом 8×8, требования, предъявляемые к алгоритму, в основном не могут быть удовлетворены.

А теперь обратимся к тому случаю, когда цель игры иная; выясним, как обстоит дело, когда типовая цель игры — выигрыш материала. Рассмотрим снова выполнение всех четырех требований.

1. Ясно, что цель эта неточная, паллиативная (так же, как и в случае цели по Шеннону). Точная цель состоит не в выигрыше вообще, а в выигрыше короля. Основное преимущество этой простой цели (выигрыш материала) состоит в том, что она связана не только со всей доской (массивом 8×8), но непременно и с полями доски. Это позволяет иметь цели игры как для совокупности элементарных систем (цель, связанную с массивом 8×8), так и для элементов совокупности (цели, связанные с отдельными полями массива 8×8). По типу цели всей системы и отдельных ее элементов одинаковы, по конкретной форме—различны. Позднее читателю станет известно, что кроме отдельных полей и всего массива

8×8 есть еще зоны игры, охватывающие некоторый комплекс полей и фигур. В зонах также есть свои конкретные цели игры того же типа, что для отдельных полей и для массива 8×8. Таким образом, для того алгоритма, который изложен далее, игра в шахматы будет не двухступенчатой, а по меньшей мере трехступенчатой системой управления. У отдельных фигур, объединенных зонами, и у фигур всего массива свои индивидуальные цели игры, но по типу общие.

Каждая фигура данного цвета стремится поразить фигуры другого цвета. Объект нападения занимает фиксированное поле массива 8×8, поразить эту мишень можно по траектории, проходящей через определенные поля доски. Раз появляется траектория, то движения фигуры становятся с точки зрения кибернетики осмысленными, появляется возможность выделять для перебора ходы, имеющие смысл, и отказаться от полного перебора ходов. Так удовлетворяется первое требование, которое было предъявлено к алгоритму.

2. Ограничение счета вариантов можно производить на принципиально иной основе. Прежде всего разъясним, в чем же состоит принципиальное различие.

Шахматная игра воспринимается шахматным мастером как неточная задача; подобные задачи люди решают непрерывно. В моей книге «Алгоритм игры в шахматы» было дано определение неточной задачи. Это такая задача, точное решение которой связано со столь большим объемом перерабатываемой информации, что устройство по переработке информации системы управления не может ее (перерабатываемую информацию) осилить. Таким образом, «точность» и «неточность» задачи определяются не только самой задачей, но и «способностями» устройства по переработке информации.

Неточная задача уже не может быть решена точно, она может быть решена лишь неточно, но как? Обычно в шахматных программах, когда цель игры определялась по Шеннону, задача решалась точно до тех пор, пока еще оставались ресурсы у ЭВМ (время и память), далее решение хирургически прерывалось, это и вызывало неточность решения.

Между тем можно еще до решения (и, видимо, это разумно) заменить задачу упрощенной моделью и это неточное отображение задачи решить более точно. В этом случае задача ограничивается не в конце пере-

бора вариантов, а до и в процессе перебора и неточность отображения выбирается такой, чтобы решение задачи стало под силу устройству по переработке информации данной системы управления и в то же время было более высокого качества. При этом принудительного обрыва счета вариантов, как правило, нет; варианты здесь разной длины и заканчиваются сами по себе (рис. 2), когда выясняется, что уже проиграно больше материала, чем может быть выиграно.

Задача ограничивается горизонтом, «дальнозоркостью» (если так можно выразиться) устройства по переработке информации. Горизонт этот в процессе игры, в процессе работы системы управления может меняться — это зависит от возникающей ситуации. Если ситуация сложная, горизонт уменьшается. Устройство же по переработке информации загружено одинаково.

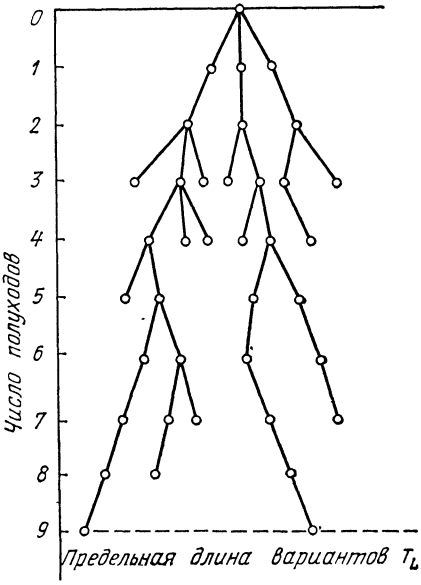

Рис. 2. Дерево перебора ходов (варианты перебора).

Таким образом, если ранее в шахматных программах задачу как бы решали точно и неточность заключалась в раннем прекращении решения, то теперь есть надежда создать неточное отображение шахматной позиции, которое может быть решено по возможности точно. И, видимо, это произойдет при удачном определении типовой цели игры.

Обрыв вариантов здесь будет принципиально иным, но в некоторых случаях, когда требуется вариант исключительно большой длины, придется ввести ограничение длины варианта (см. рис. 2), так действует и человек.

Когда я играл в 1938 г. известную партию с Капабланкой в Роттердаме, комбинация требовала расчета оптимального варианта на 24 полухода — для меня это

оказалось непосильным. Пришлось разбить оптимальный вариант на две части. Сначала я сосчитал оптимальный вариант на 12 полуходов; этот вариант можно было оборвать и пойти на него, поскольку был гарантирован ничейный исход вечным шахом. Когда же в процессе игры мы подошли к заключительной позиции первой части варианта, я сосчитал вторую часть оптимального варианта (также на 12 полуходов) и партия была закончена.

Видимо, такое же ограничение необходимо наложить и на работу программы. Это ограничение вряд ли будет фиксированным: оно должно меняться в зависимости от сложности позиции аналогично тому, как меняется предельный горизонт нападения H_L (о чем еще будет речь далее).

Здесь уместно провести параллель между практической партией и этюдом. В практической партии возможно и реально ограничение длины вариантов, а в этюде это исключено. В сложных этюдах варианты обязательно досчитываются до конца и это возможно, поскольку решение происходит при передвижении фигур на доске; в этом случае каждая возникающая позиция рассматривается как исходная (для расчета вариантов).

Теперь уже можно в общих чертах определить основное различие между данным алгоритмом и алгоритмом, основанном лишь на сокращении длины вариантов в дереве перебора. Как уже известно читателю, и в том и в другом алгоритме варианты могут иметь ограниченную длину. Может создаться ложное впечатление, что различие между алгоритмами стирается еще и потому, что в некоторых алгоритмах, основанных на полном переборе, фактически полного перебора нет: применяются различные методы, которые позволяют исключать из дерева перебора некоторые его ветви. Это сокращенное дерево перебора соответствует не истинной позиции на доске, а какой-то условной модели позиции. Но что это за модель? Это случайная модель! Вот теперь и ясна разница между двумя типами алгоритмов: качество исходной модели для анализа.

3. Если при определении цели игры по Шеннону стандартная операция передвижения фигуры предельно примитивна (траектория передвижения фигуры в один ход), ибо передвижения фигур хаотичны, то в нашем алгоритме передвижения фигур целеустремлены, направлены к определенному полю массива 8×8, осмыслены. Стан-

дартной операцией становится определение многоходовой траектории, длина которой соответствует дальности горизонта, т. е. сложности изучаемой позиции. Таким образом, стандартная операция передвижения фигуры усложняется — в этом существенная выгода. Итак, и здесь решающее значение имеет цель игры. При этом (как уже отмечалось ранее и как будет разъяснено далее) эти траектории получаются просто с использованием массивов 15×15.

4. Видимо, можно будет организовать работу нашей программы так, что (если это выгодно с точки зрения использования машины) пересчитывать заново уже сосчитанное не придется. Суть дела заключается в том, что математическое отображение позиции, состоящее из траекторий (точнее, пучков этих траекторий) в процессе игры меняется медленно. Из всего множества фигур в данный момент двигаться может только одна фигура; математическое отображение обладает большой инерционностью и поэтому пересчитывать надо лишь некоторую часть отображения. То же, вероятно, относится и к другим элементам информации, перерабатываемой ЭВМ по нашей программе.

ОБ АЛГОРИТМЕ ШАХМАТНОЙ ИГРЫ

Для иллюстрации методов применения общих принципов, изложенных выше, перейдем к рассмотрению предложенного автором алгоритма игры в шахматы.

Основные термины, применяемые в данной книге, определены в приложении 3.

Читатель уже знает, что неточная цель игры — это главное. Как уже было отмечено, в данном алгоритме цель игры — выигрыш материала. Это типовая цель; так как система управления, как минимум, трехступенчатая, то у элементов могут и должны быть разные конкретные индивидуальные цели. Кроме того, индивидуальная цель должна быть у совокупности элементов, и эта конкретная цель игры может вступать в противоречие с конкретными целями игры элементов. Подразумевается, что цель игры совокупности приводит к принятию решений, обязательных для элементов, хотя каждый элемент может стремиться и к другому решению, соответствующему его цели!

Шахматная игра имеет свои правила, которые должны быть включены в алгоритм: это и ходы фигур, и их сила (средняя стоимость) *, и превращения пешки, и то, что игра прекращается при достижении точной цели игры (выигрыша короля — мата или отсутствия ходов — пата) и т. д. Эти правила вступают в действие только при соответствующей ситуации на доске и легко фиксируются.

Особо следует сформулировать правила передвижения фигур, так как передвижения фигур — самое главное в шахматах. Правила передвижения фигур непрерывно связаны с игрой и действуют перманентно. Именно с правилами передвижения может быть связан большой объем работы устройства по переработке информации.

Важно таким образом составить эти правила, чтобы работа, связанная с перемещением фигур, была минимальной. Здесь можно воспользоваться тем обстоятельством, что передвижение фигуры данного типа с одного поля свободной от фигур доски на другое является стандартной, повторяющейся операцией. Выполнять эту стандартную операцию простым, детально отработанным приемом особенно выгодно ввиду ее сложности.

Индивидуальная цель фигуры — выигрыш материала, т. е. поражение фигуры неприятельской стороны. Выигрыш материала связан с передвижением фигуры от начальной позиции к цели (мишени), т. е. с появлением траектории движения фигуры. Определение такой траектории является важнейшей стандартной операцией, и эта операция должна найти в алгоритме простое решение.

Некоторые математики полагали, что машина в отличие от человека не может легко находить подобные траектории, что ЭВМ способна найти траекторию всего лишь в один ход. Это было заблуждением, что и станет ясно из дальнейшего.

СТАНДАРТНАЯ ОПЕРАЦИЯ ОПРЕДЕЛЕНИЯ ТРАЕКТОРИИ

Так как на свободной от фигур доске траектория (или, точнее, пучок траекторий) передвижения фигуры данного типа от заданного начального поля до другого

---

* В данном алгоритме принята следующая стоимость фигур: Кр—200, Ф—9, Л—5, С и К по 3 П—1.

заданного поля никогда не меняется, то определение такой траектории — операция не оригинальная, а стандартная. В этом и состоит возможность простого решения задачи.

Препятствием для решения является то, что исходное поле, с которого фигура может начать движение, имеет 64 значения. Напрашивается такой путь: иметь дело не с массивом 8×8 (доской) для запоминания всех возможных траекторий, а с каким-то другим массивом, где исходное поле фиксировано. Тогда, совмещая массив 8×8 с новым массивом так, чтобы начальное поле траектории в массиве 8×8 совпадало с упомянутым фиксированным полем нового массива, можно передать необходимую информацию с полей нового массива на поля массива 8×8. Нетрудно понять, что таким новым массивом должен быть массив 15×15, а фиксированным полем, с которым совмещается начальное поле траектории передвижения, должно быть центральное поле массива 15×15. Именно при этих размерах нового массива всегда возможно наложение массива 8×8 на массив 15×15 (при совпадении его центрального поля с начальным полем траектории). При меньших размерах часть массива 8×8 могла бы выходить за пределы нового массива, а при бóльших — появилась бы избыточная информация, так как часть плоскости массива, выходящая за размеры 15×15, никогда бы не использовалась. Это становится ясным при рассмотрении примерного совмещения массивов 8×8 и 15×15, изображенного на рис. 3.

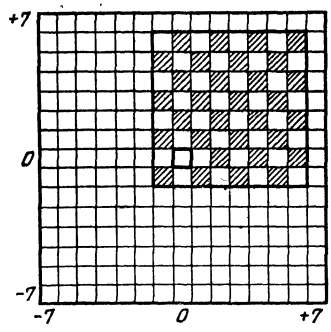

Рис. 3. Совмещение двух массивов.

Массивы 15×15 для различных фигур составляются аналогично, поэтому достаточно рассмотреть два типичных массива: для фигур, перемещающихся в одно передвижение на фиксированное расстояние (конь и, как правило, король, и пешка), и для фигур, перемещающихся в одно передвижение на разные расстояния (ферзь, ладья и слон). Поля, на которых фигура останавливается, будем называть α-полями, поля, которые

фигура проходит без остановки — β-полями. У фигур, перемещающихся на фиксированное расстояние, β-полей нет (кроме пешки, когда с начальной позиции она перемещается на два поля, и короля, когда он совершает рокировку). Массивы 15×15 для коня и ладьи изображены на рис. 4 и 5 соответственно.

На центральном поле массива фигура занимает исходную позицию. Любое другое поле массива может

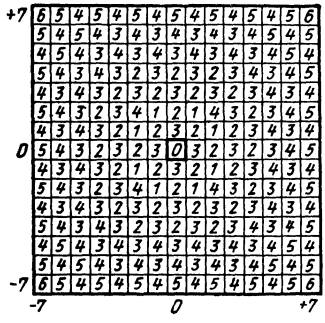

Рис. 4. Массив 15×15 для коня.

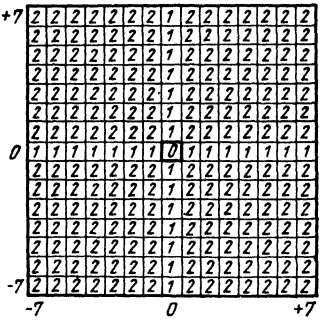

Рис. 5. Массив 15×15 для ладьи.

быть конечным полем траектории перемещения фигуры. Числа, расположенные на полях массива, равны минимальному числу ходов, необходимых для перемещения фигур по кратчайшим траекториям.

Совмещение массивов 15×15 и 8×8 эквивалентно элементарному пересчету координат. Цель совмещения состоит в том, чтобы число с поля массива 15×15, с поля, имеющего в этом массиве заданные координаты, передать полю массива 8×8, также имеющему в массиве 8×8 заданные координаты. Разность между координатами центрального поля массива 15×15 и координатами поля, содержащего передаваемое число, всегда равна разности между координатами начального поля траектории и поля, которому число передается (в массиве 8×8). Поэтому, пересчитав координаты полей массива 8×8 на координаты массива 15×15, можно легко найти искомое передаваемое число. Когда нужно передать одно или несколько чисел, операцию совмещения (передачи и запоминания всех 63 чисел) вряд ли следует производить, так как это связано с непроизводительным

трудом и расходом машинного времени. Как будет ясно из дальнейшего, получение любой траектории, в основном, связано с операцией совмещения массивов, производимой в различных комбинациях.

Цель игры была уже определена ранее, сейчас объясним способ выделения информации первостепенной важности (получение траектории). Но как быть с ограничением задачи — ограничением счета числа вариантов? Ограничение задачи, как это будет рассмотрено далее, прежде всего производится по методу горизонта, по количеству полуходов, которые должны быть затрачены на передвижение фигуры по траектории нападения на свободной от фигур доске. Чем меньше горизонт, чем короче траектории нападения, тем сильнее ограничение задачи. Итак, горизонт (будем обозначать его буквой H_L) задается в полуходах.

Поскольку одному и тому же горизонту H_L могут соответствовать предельные траектории разного числа передвижений (число α-полей при том же горизонте может отличиться на единицу, все зависит от очереди хода), то прежде всего надо определить число передвижений A, соответствующее заданному горизонту H_L. Эта элементарная операция, не требующая каких-либо пояснений. Теперь можно приступить к поиску траектории, поскольку число передвижений уже известно.

Прежде всего, необходима процедура, которая состоит в совмещении массивов 15×15 и 8×8 (при этом выбирается массив 15×15, принадлежащий той фигуре, траектория передвижения которой определяется) таким образом, что начальное α_0-поле траектории массива 8×8 совмещается с центральным полем. Операция совмещения по сути дела состоит в том, что отмечаются все координаты полей массива 8×8, на которые соответственно переходят числа 1, 2,..., $A-1, A$. Массив 8×8, у которого α_0-поле совпадает с центральным полем массива 15×15, будем обозначать $8\times8\alpha_0$. Когда же у этого массива искомые координаты полей уже определены, возьмем это обозначение в скобки: $(8\times8\alpha_0)$. Этих же обозначений будем придерживаться во всех аналогичных случаях.

Далее, следует убедиться в том, что такая траектория в условиях данного алгоритма действительно существует. Идет поиск не вообще траектории, а конкретной траектории от α_0-поля до $\alpha_к$-поля, от начального до

конечного поля возможной траектории. Если $n > A$ (n — число, которое должно быть передано $\alpha_к$-полю массива 8×8 от соответствующего поля массива 15×15), то в пределах заданного горизонта H_L траектории, очевидно, нет. Однако может быть и другой случай, когда $n < A$, а траектории (по данному алгоритму) также нет.

Суть дела в том, что для дальнобойных фигур в массивах 15×15 не содержится чисел более двух. Как известно, дальнобойными фигурами является ферзь, ладья и слон. В массиве 15×15 для ферзя даже число два встречается редко, поэтому надлежит вообще отказаться от специального массива 15×15 для ферзя, а пользоваться для нахождения траектории передвижения ферзя массивами 15×15 для ладьи и слона: там число два встречается чаще. Но даже и при введении этой тонкости нет возможности определить кратчайшую траекторию передвижения дальнобойной фугуры, скажем, в три передвижения.

Эти же трудности возникают и при определении некратчайшей траектории как для дальнобойной, так и для недальнобойной фигуры (короля и коня). Очевидно, что, когда речь идет о траектории передвижения дальнобойной фигуры в три передвижения, также подразумевается получение некратчайшей траектории. Когда мы говорим «некратчайшая» траектория, то имеем в виду некратчайшую траекторию на свободной от фигур доске. На заставленной фигурами доске эта траектория может быть уже кратчайшей. Поэтому ЭВМ должна уметь находить некратчайшие траектории.

Так как способ совмещения массивов 15×15 и 8×8 приводит лишь к определению кратчайших траекторий на свободной от фигур доске, то неизбежно приходится обратиться к поиску стыкованных траекторий, состоящих из ряда кратчайших. Чтобы не усложнять чрезмерно программу, ограничим число кратчайших траекторий, образующих стыкованную траекторию. Принято, что стыкованные траектории могут быть образованы лишь из двух кратчайших. Тогда, если ввести обозначение n_L — наибольшее число, содержащееся в массиве 15×15, то для дальнобойных фигур стыкованная траектория может состоять не более как из четырех передвижений, а вообще говоря, из $2n_L$ передвижений. Однако ничего особо плохого в этом нет. У дальнобойных фигур траектории более ограничены, нежели у недальнобойных, но

в этом есть свой смысл: действия фигур будут более координированы (пехота поспевает за танками).

Определить стыкованную траекторию, составленную из двух кратчайших, нетрудно.

Теперь можно понять, почему иногда при $n<A$ траектории все же нет: это происходит, когда $2n_L<A$. Стало быть, траекторию по данному алгоритму можно найти только в том случае, если $n\leqslant A\leqslant 2n_L$. Если это условие не выполняется, траектории нет, если выполняется, работа по поиску траектории продолжается. Тогда происходит операция совмещения, полностью аналогичная приведенной ранее; различие состоит лишь в том, что с центральным полем массива 15×15 совмещается не α_0-поле траектории в массиве 8×8, а $\alpha_к$-поле.

В массиве $(8\times8\alpha_0)$ и массиве $(8\times8\alpha_к)$ мы имеем отобранные поля (координаты полей) с соответствующими числами. Там, где на соответствующих полях (с одинаковыми координатами) сумма чисел равна n, и находятся α-поля искомой траектории (точнее, пучка искомых траекторий).

Теперь надо проверить, какого типа траектория подлежит поиску: кратчайшая или стыкованная из двух кратчайших. Если $n=A$, надлежит искать кратчайшую траекторию, что сравнительно легко; если $n<A\leqslant 2n_L$, траектория должна быть стыкованной. На этом закончим пояснение способа получения траекторий (более подробно метод изложен в приложении 1).

Итак, можно подвести некоторые итоги. Стандартная операция получения траектории передвижения фигуры по данному алгоритму (когда правильно определена паллиативная цель неточной игры) реализуется относительно простым методом. Эта операция формализуема, т. е. может быть запрограммирована. Когда получение любой траектории оказывается результатом работы подпрограммы, это означает, что ЭВМ может уже не заниматься непроизводительным перебором возможностей, лишенных смысла. ЭВМ «понимает», в каком направлении надо передвигать фигуры для реализации цели игры. Это первый робкий шаг на пути создания искусственного интеллекта, робкий, но может быть самый трудный: решена задача как выделения информации первостепенной важности, так и стандартизации сложной операции получения траектории.

ЗОНЫ ИГРЫ

Получение траектории — первая осмысленная операция, которая соответствует цели игры. Следующим шагом в этом же направлении является получение зоны игры.

Цель фигуры (элементарной системы) — уничтожение неприятельской фигуры на $\alpha_к$-поле траектории нападения. В процессе передвижения атакующей фигуры к своей мишени неприятельские фигуры противодействуют нападению, свои — поддерживают. (Здесь уже руководит ими не своя цель игры; они подчиняются решению совокупности элементарных систем, решению, соответствующему цели игры этой совокупности). Все они действуют по своим траекториям. Таким образом, часть фигур, находящихся на доске, по своим траекториям участвует в этом местном бою. Все эти фигуры и траектории их передвижения составляют *зону нападения*. Будем называть траекторию α_0-фигуры комлевой траекторией зоны.

Зону можно также рассматривать как два противоборствующих лагеря фигур; совокупность всех противоборствующих фигур на доске, участвующих в игре, можно считать совокупностью зон.

Структура зоны детерминирована. Фигуры, принадлежащие к тому же лагерю, что атакующая, будем обозначать знаком (+), фигуры того лагеря, к которому принадлежит атакованная,— знаком (—). Эти же обозначения * распространяются на так называемую связанную зону (о чем речь будет далее). Но там атакующая фигура (+) атакует не неприятельскую фигуру, а определенное поле доски; фигуры (—) противодействуют этой атаке. Фигуры (+) и (—) не белые и черные фигуры. И белые и черные фигуры могут быть одновременно фигурами и (+), и (—), но в разных простых зонах. В дальнейшем будет введено понятие сложной зоны, являющейся объединением простых. В такой зоне фигуры одного цвета могут быть и (+) и (—).

Принято, что поскольку комлевая фигура (+) атакует, в зоне игры из всех фигур (+) может двигаться только одна комлевая фигура: она пробивается к цели. Другие фигуры (+) имеют право двигаться лишь тогда,

---

* Знак (+) будем читать как «плюс»; знак (—) — как «минус».

когда они в процессе игры в зоне могут непосредственно уничтожать неприятельские фигуры (—). Отсюда ясно, что фигуры (+), которые контролируют поля какой-либо траектории в зоне, должны находиться в засаде на расстоянии одного передвижения от этого поля. Только в этом случае контролирующая фигура (+) может быть включена в зону игры.

Еще более строг отбор фигур (+) для участия в блокаде. Блокирующая фигура (+) участвует в зоне игры лишь тогда, когда она уже блокирует данное поле траектории, т. е. можно сказать, что число передвижений, приводящих к возникновению блокады, в траектории блокады равно нулю.

Фигуры (—) могут передвигаться в зоне без подобных ограничений, да это и понятно. Если цель фигур (+) в зоне состоит в том, чтобы пробиться к цели, и поэтому перемещается лишь комлевая фигура, то цель фигур (—) состоит в противодействии, а оно может быть реализовано с помощью передвижений любой фигуры (—), поэтому и может двигаться любая фигура (—), включенная в зону.

Атакованная фигура (—) может, конечно, и отступать. Для упрощения задачи принято, что траектория отступления состоит из одного передвижения. Принято также, что траектория деблокады поля тоже состоит лишь из одного передвижения. Деблокада любой фигурой, как и отступление, составляет отдельную связанную зону.

Вообще, можно принять, что в алгоритме есть только два действия фигур: нападение и отступление. Под нападением следует понимать как собственно нападение, так и контроль (в этом случае атакуется α-поле, но подразумевается, что на этом поле как бы находится или будет находиться фигура), а также блокада. Правда, может блокироваться и β-поле, где формально фигуры нет, но если принять, что дальнобойная фигура, перемещаясь от α_i-поля к α_{i+1}-полю, последовательно занимает все β-поля между этими α-полями, то и блокада всегда сводится к нападению. Под отступлением следует понимать как собственно отступление, так и деблокаду. Такая формализация облегчает работу над программой формирования зоны игры.

Теперь для иллюстрации рассмотрим нулевую зону игры, изображенную на рис. 6; эта зона нападения обве-

дена жирным пунктиром. Фигура (+, 0.0) является комлевой: она находится в комле нулевой траектории нападения. Первый нуль означает, что фигура эта принадлежит нулевой зоне (зоне нападения), второй нуль — что фигура принадлежит нулевой траектории. Стрелками указаны направления движения фигур по траекториям.

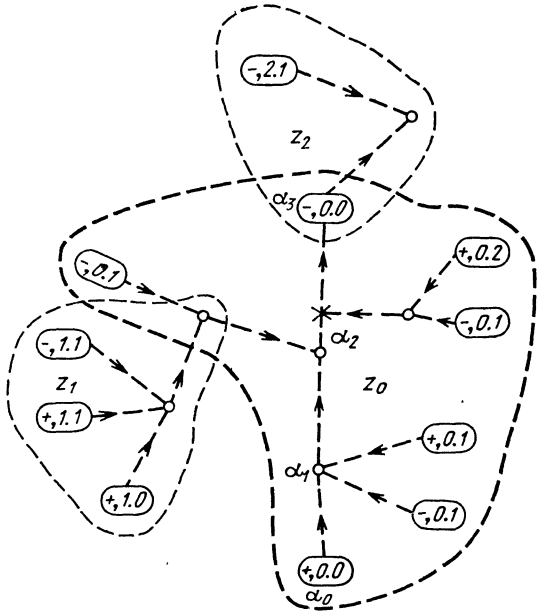

Рис. 6. Нулевая зона и две связанные.

В данном случае траектория нападения состоит из трех передвижений. На конечном $\alpha_к$-поле (в данном случае это α_3-поле) находится атакованная фигура (—, 0.0). Первый и второй нули говорят о принадлежности этой фигуры к нулевой траектории нулевой зоны.

Все остальные траектории, концы которых связаны с полями нулевой траектории, являются траекториями первого отрицания.

Траектории фигур (—, 0.1) и (+, 0.1), контролирующих α_1-поле нулевой траектории, — траектории первого отрицания, о чем и свидетельствует единица. Это же относится и к фигуре (—, 0.1), связанной с α_2-полем нулевой траектории; траектория контроля

этой фигуры состоит из двух передвижений. Наконец, есть фигура (—, 0.1), которая стремится блокировать β-поле (между α_2- и α_3-полями нулевой траектории), обозначенное крестиком; эта фигура действует по траектории первого отрицания. А вот фигура (+, 0.2), контролирующая α_1-поле этой последней траектории блокады, действует уже по траектории второго отрицания.

О горизонте для траектории нападения было уже сказано вполне достаточно, особо остановимся на горизонте траекторий первого (и выше) отрицания. Прежде всего, число передвижений здесь зависит от очереди хода. Дело в том, что горизонт нападения выгодно выбрать четным (2 полухода, 4 полухода и т. д.); в этом случае число передвижений в траектории не зависит от очереди хода, как это следует из формулы, определяющей число передвижений:

$$A = \mathrm{E}\left(\frac{H+1}{2}\right),$$

где $\mathrm{E}(z)$ — целая часть числа z.

Например, при $H=6$ или $H=5$ предельное число передвижений всегда равно трем; стало быть, если при ходе стороны (+) задать $H=6$, то, хотя при перемене очереди хода сторона (+) будет иметь лишь 5 полуходов, траектория нападения в три передвижения остается в пределах «видимости». Что же касается горизонта траекторий отрицания, то на все траектории отрицания стороны (—), связаные с данным полем траектории нападения, дается определенное число полуходов, зависящее от индекса этого поля траектории нападения. Если исходить из числа полуходов H_0, необходимых на перемещение атакующей фигуры до конца траектории нападения, и номера поля i, то на контроль α_i-поля или на блокаду β_{ij}-поля (i — номер α-поля, а j — номер β-поля из числа β-полей, следующих непосредственно за α_i-полем), можно затратить не более H_x полуходов:

$$H_x = H_0 - 2(\mathrm{к}-1)+1,$$

где к — номер конечного $\alpha_\mathrm{к}$-поля нулевой траектории.

Это количество полуходов H_x отводится не только на передвижение фигуры (—) по траектории первого отрицания, но, как уже отмечалось, и на передвижения всех отрицающих фигур (—), связанных с контролем (блокадой) данного α-поля (β-поля) нулевой траектории. Поэ-

тому после подсчета величины H_x следует вести учет расходования полуходов для того, чтобы можно было всегда вычислить предельное число передвижений для какой-то следующей промежуточной траектории.

Из формулы, определяющей H_x, следует, что H_x всегда нечетно при четном H_0. В этом случае число передвижений зависит от очереди хода. Поэтому предельное число передвижений в траекториях отрицания меняется и зона как бы «дышит»: часть отрицающих фигур то входит в зону игры, то уходит из зоны. Проще всего сказать, что есть два пограничных значения зоны, соответствующих очереди хода той и другой стороны. Нулевая зона, представленная на рис. 6, составлена для очереди хода стороны (+).

Представим себе, что размен на α_1-поле в пользу стороны (+), блокада β-поля (за α_2-полем) невозможна (это очевидно), отступление также исключено, ибо поле, куда фигура может отступить, контролируется фигурой (+). Но атакующая фигура (+) не может пробиться к цели, так как α_2-поле контролируется фигурой (—). Ситуация в зоне может быть улучшена для стороны (+) лишь при условии подведения в зону других фигур (+): вот мы и подошли к необходимости составления связанной зоны контроля. К этому еще вернемся, а сейчас отметим лишь, что программа должна начать поиск фигур, которые могут изменить положение на полях траектории первого отрицания, контролирующей α_2-поле траектории нападения.

Примем, что такая фигура найдена: это фигура (+, 1.0), которая становится комлевой фигурой связанной зоны (зоны первой степени). Мишенью этой комлевой фигуры (+) является уже не фигура (—), а α_1-поле траектории первого отрицания. Таким образом, комлевая фигура связанной зоны так же, как отрицающие фигуры, подчиняется уже не своей цели, а цели игры совокупности играющих в зоне фигур.

В данной сложной зоне исход борьбы зависит только от результата размена на α_1-поле нулевой траектории связанной зоны. Если размен в пользу стороны (+), то комлевая фигура может занять это α_1-поле и осуществить контроль над α_1-полем траектории отрицания нулевой зоны. Тогда эта комлевая фигура (+) войдет в нулевую зону как фигура (+), действующая по траектории второго отрицания нулевой зоны. Ситуация

в зоне изменится, и выигрыш материала стороной (+) обеспечен.

Здесь мы имели сложную зону, объединяющую нулевую зону и связанные. Возможны сложные зоны и другого типа. Если, например, комлевая фигура нулевой зоны сама является объектом нападения, атакованной фигурой в другой нулевой зоне, то эти две нулевые зоны также образуют сложную зону.

В рассмотренных зонах каждая фигура имела лишь одно действие, одну траекторию. Фактически бывает и иначе. Совокупность зон образует математическое отображение шахматной позиции (МО).

С помощью зон можно реализовать одно из основных требований, предъявляемых к алгоритму,—не пересчитывать уже сосчитанного. Суть дела в том, что всегда есть зоны (сложную зону также будем называть зоной), которые не зависят друг от друга. Когда по каким-либо причинам зона меняется, это не означает, что меняются все зоны игры; наоборот, это означает, что в какой-то части МО зоны остаются неизменными, а, стало быть, там и пересчитывать нечего. Реализация этого принципа, когда это выгодно, дает возможность программисту сделать работу машины производительней.

Фигуры и их траектории еще не являются элементами МО, это только материал, из которого МО строится. Зоны — это уже составные элементы МО.

Итак, мы убедились, что наш алгоритм удовлетворяет четырем принципам составления алгоритма: выделению информации первостепенной важности, ограничению счета вариантов перебора (хотя это еще подлежит дальнейшему рассмотрению), стандартизации возможно более сложных повторяющихся операций и тому, чтобы не пересчитывать уже сосчитанного.

ФОРМИРОВАНИЕ ЗОН И ПЕРЕБОР. ПСЕВДОПЕРЕБОР

Теперь, когда мы познакомились с понятием зоны игры, можно перейти к формированию зон и тем самым МО.

Все начинается с формирования нулевых зон нападения, когда МО создается заново. В пределах горизонта

H_L определяются все комлевые траектории нападения; это единственная операция при формировании зоны, не зависящая от перебора. Все дальнейшие операции, по предложению Б. Штильмана, идут параллельно с перебором: происходит одновременно и перебор, и формирование зон, и оценка вариантов. Такой метод работы программы наиболее экономичен, ибо, как только оценка определит окончание варианта, перебор, связанный с этим вариантом, а, следовательно, и формирование зоны, прекращаются. Это позволяет свести к минимуму объем как сформированной зоны игры, так и перебора.

Можно напомнить, что нулевая зона игры создается с траекториями фигур (—) в горизонте H_x и с траекториями фигур (+) из одного передвижения; траектория отступления фигуры (—) с $\alpha_к$-поля состоит из одного передвижения. Как уже отмечалось, принцип, по которому определяется горизонт H_x прост: принято, что сторона (+) в зоне имеет право двигать лишь свою комлевую α_0-фигуру (поэтому все остальные фигуры этой же стороны находятся от своих конечных полей на расстоянии не более одного передвижения, они уже готовы к бою), а сторона (—) включает в зону лишь те свои фигуры, которые могут поспеть к тому или иному полю, чтобы вовремя помешать продвижению какой-либо фигуры.

Следует также помнить, что в общем случае мы имеем дело не с траекториями, а с пучками траекторий; пучки эти состоят из траекторий разной длины, α_0- и $\alpha_к$-поля являются общими для всех траекторий пучка (у траекторий пучков отступления или деблокады в одно передвижение общим является лишь α_0-поле). В дальнейшем будем говорить «траектория», а подразумевать «пучок траекторий».

Таким образом, до начала перебора мы имеем лишь множество комлевых траекторий нападения, у которых на α_0-полях расположены нападющие фигуры и на $\alpha_к$-полях — атакованные. В первую очередь, должны быть рассмотрены «вилочные» (частично совпадающие) траектории, ибо когда фигура движется по совпадающим траекториям, она перемещается во столько раз «быстрее», сколько траекторий совпадают.

При наличии вилочности игра ускоряется. Каждая вилочная траектория имеет обязательно область невилочных частей (несовпадающих), состоящих из раз-

ного числа передвижений, и вилочную область, общую для совпадающих траекторий.

В некоторых случаях выгоды, которые можно получить от вилочности траекторий, формализуются. Например, примем, что невилочные части вилочных траекторий таковы, что фигура, находясь на последнем α-поле вилочной части траектории, успевает принять участие в игре, если совершит еще один ход, передвигаясь по любой из невилочных частей. Обозначим число таких невилочных частей траектории через n. Тогда, если вилочная часть траектории состоит из числа передвижений, не превосходящего $n-1$, то фигура непременно успеет принять участие в игре, хотя бы по одной из невилочных частей траектории. Программа должна учитывать такое возможное удлинение вилочной траектории (хотя и здесь предельный горизонт H_L не должен быть превзойден), отдавая предпочтение этим траекториям.

Знаменитый пешечный этюд Рихарда Рети (Белые: Крh8, Пс6. Черные: Кра6, Пh5. Белые начинают и делают ничью) на заключительном этапе решения иллюстрирует этот принцип удлинения вилочных траекторий.

После 1. Крg7 Крb6 2. Крf6 h4 мы имеем две вилочные траектории Крf6—e5—d6 и Крf6—e5—f4. Вилочная часть траектории Крf6—e5 состоит из одного передвижения; число невилочных частей равно двум (условие $n-1 \leqslant 1$, т. е. $2-1 \leqslant 1$ удовлетворяется). Путем 3. Крe5! король белых успевает принять участие в игре в той или иной зоне (либо на ферзевом фланге, либо на королевском) и ничья неизбежна.

Теперь вернемся к комлевой траектории зоны: задача заключается в том, чтобы проверить, проходимо ли α_1-поле для α_0-фигуры, каков результат борьбы, если α_0-фигура ступит на α_1-поле?

Для того чтбы определить результаты борьбы на α_1-поле, мы должны заранее знать ту часть нулевой зоны, которая связана с α_1-полем, т. е. найти траектории отрицающих фигур (+) и (—), для которых α_1-поле (а также любое β-поле между α_0- и α_1-полями) является концевым $\alpha_к$-полем. Для этого должен быть создан механизм псевдоперебора для поиска траекторий, суть которого состоит в том, что ходу фигуры в переборе должен предшествовать псевдоход, идентичный ходу этой фигуры, но являющийся фиктивным, предпринимаемым лишь для разведки.

Получение траекторий (кроме траекторий отступления и деблокады), по предложению Б. Штильмана, происходит по единому методу: когда в заданном горизонте на свободной от фигур доске одна фигура «видит» другую, то формируется траектория «нападения». Поэтому и совершается псевдоход α_0-фигурой на α_1-поле. Когда фигуры (—) в заданном горизонте видят α_0-фигуру (+) на α_1-поле, появляются отрицающие траектории фигур (—). Аналогично решается задача поиска отрицающих траекторий блокады β-полей на участке α_0—α_1: программа «передвигает» дальнобойную фигуру не прямо с α_0 на α_1-поле, а «ставит» фигуру на каждое β-поле последовательно, определяя при этом все отрицающие траектории блокады.

Когда траектории отрицания фигур (—), траектории, у которых α_1-поле является концевым $\alpha_к$-полем, известны, можно сделать «настоящий» ход α_0-фигуры на α_1-поле. Фигура (—) бьет α_0-фигуру (+), но не сразу, сначала делается псевдоход фигурой (—) на α_1-поле, определяются траектории в одно передвижение фигур (+) и, если таковые находятся, они включаются в перебор и все начинается сначала. Так и действует этот механизм псевдо- и подлинного перебора: как только псевдоперебор обнаруживает новые траектории, перебор начинается сначала. Лишь тогда, когда оценка определяет обрыв варианта и псевдоперебор не дает новых траекторий, прекращается и перебор и формирование зоны: вариант перебора определен.

ПОИСК СВЯЗАННЫХ ЗОН

Этот вопрос уже обсуждался, рассмотрим его более подробно. Как мы видели, комлевые траектории нападения появляются безусловно. Зона игры, хотя бы частично, непременно образуется и настолько, насколько это нужно. Однако бывает так, что в зону игры должна быть вовлечена другая фигура, например, тогда, когда сторона (+) недовольна достигнутым результатом. Для того чтобы найти в пределах заданного горизонта H_L какую-то фигуру (+) и ввести ее в нулевую зону игры, должна быть сформирована особая зона игры, связанная с нулевой зоной. Точнее, эта зона связана не со всей

нулевой зоной, а с конкретным полем траектории, принадлежащей нулевой зоне, с тем α- или β-полем, где ситуация (результат игры) должна быть улучшена. Это приводит к принципиальному отличию формирования связанных зон от формирования нулевых.

Нулевые зоны формируются, если нападение возможно, связанные — если они нужны, если они могут изменить ситуацию на каком-либо поле траектории. Поэтому связанная зона создается лишь в том случае, если фигура, которая будет вовлечена в игру, способна, например, изменить результат размена на α-поле. Прежде всего, проверяется, может ли эта фигура изменить результат размена (если, конечно, эта фигура находится в горизонте H_L) и лишь при выполнении этого условия идет поиск комлевой траектории связанной зоны.

Связанные зоны могут выполнять различные функции: контроля, блокады, деблокады, отвлечения — те, что требуются для улучшения борьбы.

Зоны, связанные с нулевой зоной, будем называть связанными зонами первой степени, зоны, связанные со связанной зоной первой степени,— связанными зонами второй степенни и т. д.

ФОРМИРОВАНИЕ СЛОЖНЫХ ЗОН

Простая нулевая зона не исключение, но и не правило. Если с ней связана какая-нибудь другая зона, как это было рассмотрено, то образуется сложная зона. Сложные зоны могут образовываться и иначе. Например, если есть простые зоны, имеющие общие фигуры, то вместе эти зоны также образуют сложную зону. Различие будет в том, что в сложной зоне, образованной из одной простой нулевой и из связанных с ней зон, на $\alpha_к$-поле будет лишь одна атакованная $\alpha_к$-фигура, в сложной же зоне, в которую входят несколько нулевых зон, может быть несколько $\alpha_к$-фигур.

Как же формируются сложные зоны? Как формируется МО? Основное в шахматах при выборе хода — перебор ходов, варианты ходов. Зоны, МО — это только подготовительная процедура, фундамент, на котором основывается направленный, осмысленный перебор ходов и своевременное окончание (обрыв) вариантов перебора. Вне перебора ходов зоны не имеют самостоятель-

ного значения. Более того, нет зон и МО вообще, есть зоны и МО, привязанные к каждому узлу ветви дерева перебора. С каждым ходом перебора при переходе от одного узла к другому меняется МО. Зона, в которой идет игра при данной ветви перебора ходов, отличается от зоны в другой ветви перебора. Не меняется лишь одно МО, привязанное к исходной позиции (после того, как это МО окончательно сформировано), да это и понятно, ибо у всех вариантов перебора в исходной позиции один общий узел.

Теперь можно объяснить, как формируются сложные зоны. Они формируются в процессе перебора. При одном варианте перебора сложных зон вообще может не оказаться, перебор идет в одной простой зоне, при другом — перебор идет в сложной зоне, где одно $\alpha_к$-поле и только одна мишень, в третьем — перебор идет в сложной зоне, где несколько фигур на $\alpha_к$-полях и т. д. Может быть вариант, который охватывает одновременно несколько независимых сложных зон.

Итак, имеется великое множество МО, ибо узлов перебора много. Все это надо запомнить. Справится ли с этим ЭВМ? Суть дела в том, что эти МО имеют сходство. Все они произошли от одной «матери» — МО исходной позиции. Кроме того, с каждым ходом перебора МО меняется незначительно. На этом и основывается надежда, что все это запомнить можно. Конечно, запомнить надо не каждое МО в отдельности, это непосильная задача, запомнить надо отличие данного МО от исходного. Как же это реализовать?

Здесь напрашивается следующий метод: должно быть одно общее МО, но используется оно при каждом данном варианте перебора частично. Та часть МО, те траектории, которые уже не используются в данной ветви дерева перебора, но могут быть использованы в других ветвях, должны «застыть»; когда они понадобятся для других ветвей, эти застывшие траектории должны быть «размерожены».

В процессе данного варианта перебора МО может не только сокращаться (из-за застывания), но и расширяться из-за появления новых зон, новых траекторий. Эта новая часть МО должна запоминаться, а после использования (когда вариант оценен и эта новая часть МО уже не может быть использована вообще) стираться в памяти вместе с оцененной ветвью перебора.

Таким образом, новое зонообразование (образование новых траекторий) наряду с застыванием, размораживанием и стиранием — способы, обеспечивающие индивидуализацию МО в данном узле ветви дерева перебора.

Застывание части траекторий в зоне является логичным требованием, сокращающим перебор ходов. Как только фигура прошла какое-либо α- или β-поле, часть траекторий в зоне теряет смысл. До тех пор, пока решение не принято и ход не сделан, застывание и размораживание происходят непрерывно. Как только ход сделан, часть траекторий зоны должна быть аннулирована и зона сокращена. Траектории, подлежащие аннулированию, и будут теми самыми траекториями, которые застыли бы, если бы ход был сделан не на доске, а в переборе. После того, как ход сделан, следует дополнить МО новыми нападениями, ибо границы вариантов как бы отодвинулись.

Отметим особую роль комлевых траекторий зон (нулевых либо связанных). Зона строится по частям, связанным с α_1-,...,$\alpha_к$-полями комлевой траектории. Если оптимальный вариант таков, что вариант оборван в частях зоны, связанных с полями $\alpha_1,...,\alpha_i$, то поиск части зоны, связанной с полем α_{i+1} комлевой траектории, не производится.

Это относится и к нулевой и к связанной зонам. Таким образом, связанная зона так же, как и нулевая, может быть сформирована частично, и, если связанная зона не формируется полностью, она и не связывается с той зоной, с которой она могла бы быть связана.

Сложная зона составляется из нескольких нулевых следующим образом. Формируются нулевые зоны так, как это было объяснено ранее. Затем проверяется наличие у зон общих фигур и полей; если проверка дает положительный результат, то эти зоны образуют сложную зону. Перебор, который был уже проведен в этих зонах, оказывается предварительным.

Важнейшее влияние на формирование зоны имеют новые нападения. Фигура, передвигаясь в процессе перебора, может «увидеть» в пределах заданного горизонта H_L неприятельскую фигуру; случается и обратная картина, когда неприятельская фигура в процессе перебора оказалась в поле зрения «неподвижно» расположенной фигуры. Надо ли при этом фиксировать эти но-

вые нападения, создавая новые нулевые зоны? Если фиксировать все эти новые нападения без ограничения, информация разрастается и МО расползается во все стороны. Здесь необходимо разумное ограничение.

Новые нападения могут быть двух типов: относящиеся к контролирующим или блокирующим действиям фигур в горизонте H_x и собственно нападения на фигуры в горизонте H_L. Ограничивать число контролирующих и блокирующих фигур в горизонте H_x вряд ли есть основания, ибо эта новая информация имеет локальный, быстро суживающийся характер. А вот нападения в горизонте H_L, связанные с формированием новых зон, должны быть тщательно отфильтрованы и включены лишь те, что имеют разумные основания.

Есть еще одна принципиальная разница между нападениями в горизонте H_x (контроль и блокада) и нападениями в горизонте H_L. Нападение в горизонте H_L появляется лишь тогда, когда атакованная фигура уже находится на $\alpha_к$-поле возможной траектории нападения; траектории контроля и блокады в горизонте H_x возникают до того, как данная фигура достигла контролируемого (блокируемого) поля.

Сделаем сейчас небольшое отступление и покажем, как это новое нападение в горизонте H_L может образоваться. Рассмотрим случай, когда какая-то фигура, продвигаясь по своей траектории, достигла α_i-поля; в этот момент эту фигуру в горизонте H_L «увидела» неприятельская фигура. Программа составит пучок траекторий нападения. Надо ли при этом формировать зону нападения и включать зону в МО?

Все это произошло в каком-то узле ветви перебора. Запомним пучок комлевых траекторий нападения и будем продолжать перебор, спускаясь по ветви перебора еще ниже до окончания (обрыва) варианта. Примем при этом, что никаких новых нападений уже не было. Так как же быть с тем пучком траекторий нападения, который был зафиксирован?

Найдем, сколько полуходов в текущем оптимальном варианте передвигавшаяся фигура оставалась на α_i-поле.

Если неприятельская фигура по зафиксированной траектории успеет на нее напасть, то новую зону нападения создавать, если не успеет, то, как правило (за исключением, на котором останавливаться не будем), не создавать.

Таким образом, в одном варианте перебора новую зону следует формировать, в другом — нет. Это еще раз подтверждает принцип, что МО жестко связано с вариантом перебора, точнее, с каждым узлом ветви перебора.

Если программа решит создавать зону, то происходит подъем по варианту с определением минимакса до того узла, где зафиксирована комлевая траектория нового нападения. Эта траектория включается в МО и перебор продолжается: возобновляется спуск по новым ветвям перебора уже с включенной в перебор новой зоной нападения.

Наконец, мы вновь поднялись по варианту к этому узлу перебора; теперь уже все исследовано, новые нападения ниже этого узла проверены, минимакс дал свои результаты. Надо подниматься выше по варианту, выше того узла, ниже которого мы исследовали влияние этого нового нападения. Что же теперь делать с зоной нападения, которая появилась в этом узле? Больше она никому не нужна: она повлияла (или не повлияла) на оценку и сделала свое дело. Информация о зоне должна быть стерта, так же как и информации о пройденной ветви перебора (разумеется, к текущему оптимальному варианту это не относится). Однако информацию о пучке нападения пока сохраним!

Продолжим подъем еще выше по варианту, дойдем до какого-то узла, снова начнем перебор, т. е. будем спускаться по варианту вниз до обрыва варианта. Исследуем, как в этом случае передвигалась наша фигура относительно α_i-поля, к которому привязан был пучок траекторий (информация об этом пучке все еще хранится). Выясним, что фигура в этом варианте не проходила α_i-поле. Что же делать дальше?

Мы должны решить вопрос о связанной зоне и собрать сведения о результатах размена фигур на α_i-поле. Может создание связанной зоны улучшить результаты размена на α_i-поле? Если да, зону создавать, если нет, связанной зоне не быть (пучок стереть в памяти). Создавать связанную зону следует, как только появилась траектория, на поле которой улучшается размен.

НЕЗАВИСИМЫЕ ЗОНЫ

Независимая зона (в том смысле, что она не зависит от других зон: в одном варианте перебора нет общих фигур и полей с другими зонами) может быть

простой нулевой, сложной нулевой со связанными зонами и, наконец, сложной зоной, состоящей из нескольких нулевых зон, имеющих в общем случае и связанные зоны.

Перебор в независимой зоне казалось бы не связан с переборами в других независимых зонах, так как у этих зон нет общих фигур и полей, но эти переборы могут быть связаны распределением избыточного времени, если оно есть в каких-либо независимых зонах. К определению понятия избыточного времени мы еще вернемся.

Практическая независимость переборов в независимых зонах предопределяет ограничение перебора в целом, ибо варианты ходов резко сокращаются. Сокращаются они еще и потому, что в большинстве независимых зон оптимальный вариант перебора состоит в том, что фигуры должны оставаться на своих местах.

Будем рассматривать наиболее сложную независимую зону, составленную из множества простых нулевых и связанных зон, причем эти зоны содержат как белые, так и черные α_0-фигуры. Следует определить, до каких пор продолжать вариант перебора? Обрыв варианта в каждой простой зоне, входящей в сложную, может происходить, если α_0-фигура уничтожена или заблокирована так, что снятие блокады невозможно, и если $\alpha_к$-фигура уничтожена или отступила на безопасное поле.

При выполнении одного из условий игра в этой простой зоне должна быть прекращена, что, однако, не означает прекращения игры в других простых зонах. Наоборот, игра в других зонах (простых) должна продолжаться, т. е. игра в той сложной зоне, которую мы рассматриваем, должна продолжаться до тех пор, пока не будет выполнено еще одно условие, определяющее прекращение перебора уже во всей сложной зоне. Прежде всего рассмотрим действие этого условия в независимой простой зоне, т. е. сведем наиболее сложный случай к наиболее простому. Каково тогда будет третье условие обрыва варианта?

Ранее автором был сформулирован следующий принцип: то, что потеряно, должно быть меньше того, что может быть выиграно. Это положение было названо «формулой надежды». Однако этот принцип должен быть расширен, поскольку справедливо и иное положение: можно что-то терять в надежде, что в будущем благодаря этому будет потеряно меньше. Смысл принципа заключается в том, что всегда лучше потерять сейчас меньше,

чём в дальнейшем больше! Это означает, что не следует обрывать вариант, когда сторона (—) потеряла материал меньший, чем $m_к$, ибо возможно, что это спасет $\alpha_к$-фигуру.

Итак, формулу надежды можно записать в следующем виде:

$$|\Delta m| < |m_к|,$$

где Δm — материал, потерянный любой стороной, а $m_к$ — стоимость $\alpha_к$-фигуры. Это формула относится к простой независимой зоне.

В сложной независимой зоне может быть уже не одна $\alpha_к$-фигура, а несколько таких фигур, и они могут быть разного цвета. Таким образом, вместо $m_к$ мы имеем $\Sigma m_{к\,b}$ и $\Sigma m_{к\,w}$, где $m_{к\,b}$ — стоимость черных $\alpha_к$-фигур, а $m_{к\,w}$ — стоимость белых $\alpha_к$-фигур. Как же определить допустимую величину потерянного материала в сложной зоне, когда вариант перебора еще нужно продолжать?

Ясно, что если материал, потерянный белыми в своих простых зонах, меньше $|\Sigma m_{к\,b}|$, то вариант продолжать надо. А если белые потеряли при этом материал и в «черных» зонах, что тогда? Очевидно, если этот потерянный материал меньше $|\Sigma m_{к\,w}|$, вариант также нужно продолжать. Следовательно, если

$$|\Delta m| < |\Sigma m_{к\,b}| + |\Sigma m_{к\,w}|,$$

то вариант перебора продолжается. Эта формула действует и для белых и для черных. Формула превращается в предыдущую, если сложная зона превращается в простую. Следует иметь в виду, что $\Sigma m_к$ может меняться от хода к ходу.

Таким образом, при переборе в сложной независимой зоне должен вестись двойной учет. Первые два условия определяют прекращение варианта перебора в каждой простой зоне. Третье условие обрывает вариант перебора во всей сложной независимой зоне.

ПРИНЦИП ОДНОГО ВЫИГРЫША

Для ограничения перебора информации установим принцип одного выигрыша. Суть его состоит в том, что каждая сторона готова при переборе довольствоваться выигрышем материала в одной независимой зоне при

непременном условии, что этот выигрыш достигается в той зоне, где он наибольший. Из этого принципа вытекают следствия, представляющие практический интерес. Следует начинать перебор в тех независимых зонах, где стоимость $\alpha_к$-фигур наибольшая. Если в какой-то зоне материал выигрывается, то производить перебор в зонах, где стоимость $\alpha_к$-фигур равна или меньше этого выигранного материала, вообще не нужно. И главное, играть, по сути дела, надо в одной независимой зоне. Так поступает каждая сторона, т. е. игра может идти в двух независимых зонах разного цвета. Все это способствует сокращению перебора информации.

Вероятно, в способности выделять главное и состоит одно из отличий человека от более простых существ. Например, стрекоза или лягушка так действовать не могут. Они атакуют цель, если она единственная; если цель отсутствует или их несколько, стрекоза или лягушка бездействует. Они не могут из множества целей выделить главную. Программа их такова, что действовать в этом случае им не положено. Человек умеет выделять из всей массы информации главное; это же должно быть передано ЭВМ, чтобы она не играла в шахматы так, как стрекоза атакует свою жертву...

ДВИЖЕНИЕ И ДЕЙСТВИЕ. ИЗБЫТОЧНОЕ ВРЕМЯ

На первый взгляд кажется, что фигуры могут двигаться и только, но в действительности это не так. Как человек может суетиться, но не работать, так и фигура может двигаться, но не действовать. Движение — это необходимое условие (за тем исключением, когда фигура выполняет функцию блокады), действие — достаточное условие, чтобы оценить полезную роль фигуры в игре. Поэтому все программы, основанные на полном переборе, не могут привести к достижению цели: в основном они занимаются суетой сует. В той программе, которая должна быть создана, суеты в переборе существенно меньше, но, что делать, все же она есть. Более того, когда перебор в независимой зоне закончен и в результате минимакса найден оптимальный вариант перебора, даже в этот оптимальный вариант могут включиться впустую двигавшиеся фигуры — они двигались, но не действовали. Время, израсходованное на передвижение

этих фигур, является потерянным. Несомненная выгода состоит в том, чтобы вернуть эти «фигуры-трутни» на их исходные позиции, а появившееся избыточное время использовать в других зонах (или даже в этой же зоне, но для передвижения «фигур-пчел»). Время в шахматной игре — материальный ресурс. По сути дела, речь идет о целесообразном использовании ресурсов при планировании работы.

Бездействующие фигуры появляются в оптимальном варианте потому, что у стороны (—) может не быть в этой независимой зоне полезного хода и делается любой возможный ход в зоне (соответствующий правилам игры в зоне). Задача состоит в том, чтобы заменить движение этих псевдодействующих фигур на паузы и во время этих пауз передвигать с пользой фигуры того же цвета в другой зоне.

Рациональное использование ресурса времени может повлиять на выбор хода. Вероятно, можно предложить несколько методов поиска псевдодействующих фигур в оптимальном варианте.

Рассмотрим наиболее простой. При переборе в МО включается не более двух зон (одна белая и одна черная) и не будет большого отступления от принципа ограничения перерабатываемой информации, если перебор этих двух зон (во времени) сделать совместно. Когда совершается ход фигурой (—) в первой зоне, то в другом варианте перебора в первой зоне вместо этого хода сделать паузу и сыграть какой-либо фигурой (+) того же цвета во второй зоне. Оптимальный вариант перебора, общий для двух зон при этом методе, уже не будет содержать ходы псевдодействующих фигур или эти фигуры не будут влиять на результат перебора.

ОБЩАЯ ПОЗИЦИОННАЯ ОЦЕНКА

Позиционная игра подразумевает общую позиционную оценку или, иначе говоря, дополнительную цель игры, которая должна сочетаться с оценкой материальной, с основной целью игры.

Ранее я предполагал, что та общая оценка (оценочная функция) или цель игры, которую предложил К. Шеннон (и применяют все его последователи), может быть приобщена к основной оценке (по материалу), принятой

в данном алгоритме. Когда же в процессе работы над программой основная цель должна была быть дополнена общей позиционной целью, пришлось убедиться (как об этом уже упоминалось), что оценочная функция Шеннона, заимствованная из учебников для начинающих, фальшива. В среднем, она справедлива для большинства великого множества шахматных позиций. Эта усредненная оценка фальшива (или может быть фальшива) для той конкретной позиции, которая должна быть оценена с общей точки зрения.

Действительно, учебники говорят, что сдвоенные пешки это плохо, расположенные рядом — хорошо. Но в одной позиции сдвоенные пешки (хотя и редко) могут быть позиционным плюсом, а в другой позиции расположенные рядом пешки — минусом. «Форточка» в пешечной позиции короля — дело хорошее. В некоторых же позициях наличие «форточки» приводит к непоправимому ослаблению позиции короля. Открытую линию, вообще говоря, следует занимать: это справедливо для многих позиций, для других — лишено смысла.

Напрашивается вывод, что общая оценка для данной конкретной позиции, которую ищет и находит мастер, как правило, не соответствует той оценочной функции, которая предложена Шенноном. Однако какая-то разумная общая оценка должна быть непременно. Какова же она? Она должна быть индивидуальной для данной позиции, а если это так, то основой для общей позиционной оценки может являться все то же МО, ибо оно и только оно индивидуализирует наше представление о позиции.

МО одновременно служит основой для дерева перебора ходов (по траекториям) и основой для позиционной оценки, и в этом последнем случае мы имеем дело с полями траекторий.

Если траекторий может быть превеликое множество, то число полей ограничено (не более 64). Но позиционных оценок борьбы за эти поля может быть и больше и меньше 64, ибо для каждой траектории фигуры, проходящей через какое-либо поле, существует своя оценка. Если меняются контроль, блокада или деблокада этих полей, то это приводит к изменению общего представления о позиции, к изменению оценки позиции.

Итак, позиционная цель — поля траекторий, их контроль и блокада. Фигура, участвующая в позиционной игре, должна или блокировать, или контролировать поле

на расстоянии одного передвижения — при большем расстоянии получается уже маневренная игра. Ясно, что когда контроль или блокада прочные, прочны и позиционные факторы.

Можно предложить следующую оценку, хотя истинность ее может быть проверена лишь в эксперименте. Рассмотрим все поля траекторий всех фигур, поля, которые контролируются в одно передвижение другими фигурами, и поля, которые блокируются. Подсчитаем результат размена на поле, когда на это поле ступила активная фигура. Когда размен на каком-либо поле закончен и оценен, фигура ставится на исходное в данной оцениваемой позиции поле и наступает очередь для следующей оценки размена на этом поле для другой фигуры (если поля траекторий совпадают).

Фигура может «продвигаться» по полям своей траектории для оценки результата размена, как она продвигается в вариантах перебора. Если фигура в процессе своего движения по траектории проходит α-поле, где размена нет (т. е. поле свободно для передвижения), то поле проходимо (тот же случай, когда $\Delta m > 0$). Если траектория блокирована и на α-поле не попасть, то как бы $\Delta m = 0$, поле непроходимо. Таким образом, деблокада может изменить оценку позиции существенно, если последующие α-поля также окажутся проходимы.

Очевидно, что при позиционной оценке очередь хода не имеет значения; считается как бы, что очередь хода за каждой стороной одновременно. Фиксировать надо лишь те результаты, которые получаются при активных действиях каждой стороны.

Резюмируя, можно сказать, что позиционная оценка базируется на результатах размена на полях. Подсчет этих результатов формально не связан с деревом перебора или связан с деревом перебора постольку, поскольку связаны с деревом перебора траектории.

Вновь напомним, что, кроме перемещающейся фигуры, остальные фигуры участвуют в размене, если они находятся на расстоянии одного передвижения.

С изменением МО в процессе перебора ходов позиционная оценка может меняться. Иначе говоря, каждому ходу перебора соответствует своя позиционная оценка МО, которая должна приобщаться к материальной оценке, также связанной с перебором. Перевес в позиционной оценке, как это диктуется шахматной практикой, ком-

пенсирует потерю не более двух пешек (количественные соотношения будут окончательно определены в эксперименте). При материальном равенстве решающее слово при выборе хода за позиционной оценкой.

Можно добавить, что позиционная жертва возможна, когда позиционный перевес прочен, т. е. не уменьшается при выбранном варианте перебора ходов.

Позиционная игра реализуется с помощью позиционных зон игры. Следует проводить различие между связанной и позиционной зонами. О связанной зоне все уже известно: это зона, которая непременно должна быть связана с другой зоной более низкого порядка. Позиционная же зона существует независимо от других зон. Связанная зона оказывается бесполезной, если в зоне, с которой она связана, игры нет. Позиционная зона как раз и имеет смысл, когда в зоне, где находится поле траектории, за которое идет борьба, игры нет. Обе зоны как бы совпадают, только цели у них разные.

ПОЗИЦИОННАЯ ЖЕРТВА

При позиционной оценке учитываются поля с положительным результатом размена $+1(\Delta m>0)$ и только те поля, которые участвовали в переборе. Таким образом, мы находим комплекс полей в МО, по которым фигуры могут более или менее свободно перемещаться, — это как бы плацдарм для боевых операций. Но это не то тривиальное понятие владения пространством, которое можно встретить во всех учебниках (кроме учебника Капабланки), это тот реальный комплекс важных полей (полей реальных траекторий), в котором фигуры могут успешно действовать.

Перебор дает ответ на первый вопрос задачи о перспективном планировании: что выгодно с точки зрения текущего момента? При этом определяется комплекс полей, контроль которых позволяет прогнозировать успех в перспективе. Таким образом, мы можем ответить и на второй вопрос задачи: что ждет нас в будущем?

Оценка вариантов при переборе учитывает материал и контроль полей, сегодняшний день и будущее. Еще раз отметим, что практика показывает: мастера не жертвуют больше двух единиц материала, поэтому позицион-

ная оценка не может изменить материальную более чем на две пешки.

Обозначим число контролируемых полей белыми фигурами через K_w, а черными — через K_b. Их отношение $K=K_w/K_b$ и будет позиционной оценкой. При $K \geqslant A$ принимаем $Ж \approx 2$, а при $B \leqslant K \leqslant A$ принимаем $Ж=1$, где A и B — числа, которые будут определены экспериментально, $Ж$ — жертвуемый материал. Число K показывает, во сколько раз одна сторона контролирует больше важных полей, чем другая.

ОБЩАЯ БЛОК-СХЕМА АЛГОРИТМА

Предлагаемый алгоритм иллюстрируется блок-схемой, изображенной на рис. 7.

Схема составлена для случая, когда надо принять решение об очередном ходе во время партии. Если следует

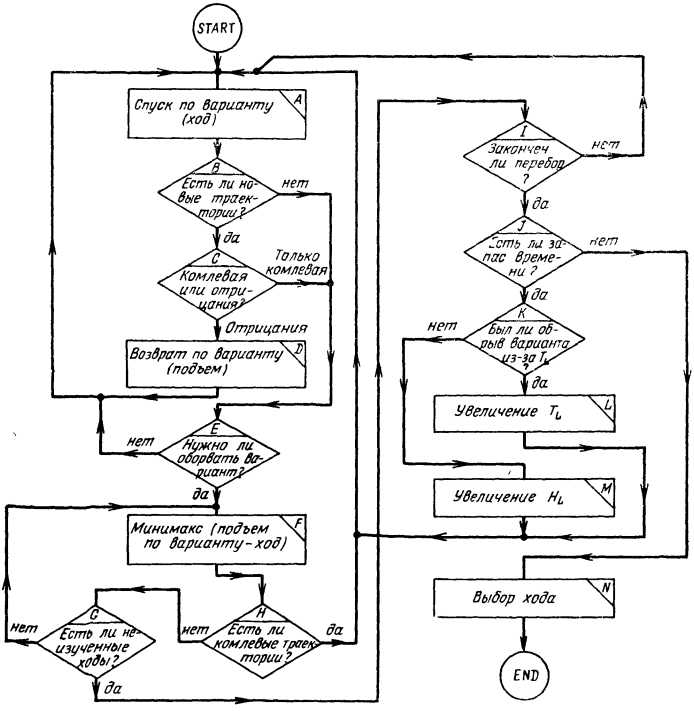

Рис. 7. Общая блок-схема алгоритма.

найти ход в новой исходной позиции, то работа программы должна начинаться с поиска траекторий нападения (в данной схеме эта процедура опущена).

Нетрудно заметить, что в схеме в явном виде нет формирования зоны (это подразумевается), ничего не говорится и о застывании, размораживании и стирании, правилах обрыва варианта и пр. Эти процедуры более или менее ясны. В схему включена наиболее тонкая часть алгоритма, состоящая в формировании новых траекторий отрицания и новых зон. Рассмотрен вопрос и о возможности увеличения предельного горизонта нападения и предельной длины варианта.

Работа программы начинается с того, что совершается один ход, включаемый в перебор (процедура A). Незамедлительно в проверке B отмечается появление новых траекторий. Если их нет, работает проверка E; если вариант не закончен, вновь работает процедура A и вариант продолжается. Если новые траектории есть, то проверка C определяет, какие это траектории — комлевые или отрицания. Если среди траекторий обнаружены траектории отрицания, то необходим возврат по варианту (процедура D), чтобы перебор начался вновь с того узла, с того момента, когда отрицающая фигура попала на α_0-поле, но не ранее того, как начато формирование зоны.

Наконец, снова работает проверка C и на сей раз обнаружены только комлевые траектории. До того, как вариант закончен, нельзя решить вопрос о целесообразности формирования новой зоны, поэтому вновь должна работать проверка E.

Примем, что вариант закончен, тогда надо начинать минимакс — работает процедура F. Надлежит проверить, обнаружены ли были ранее в этом узле перебора комлевые траектории в горизонте H_L? Если нет, то либо продолжается минимакс (процедура P), либо спуск по новой ветке перебора (процедура A) в том случае, если перебор не закончен, что определяет проверка I. Если же комлевые траектории есть и должны быть сформированы новые зоны, то включается в работу процедура A и перебор уже происходит с включением в МО новых зон.

Вернемся к проверке I. Примем, что на сей раз перебор закончен. Осталось проверить, увеличивать ли точность решения (предельную длину варианта) или предельный горизонт? Это необходимо лишь в том случае,

если есть запас времени. Увеличивать T_L следует, конечно, лишь тогда, когда ранее варианты преждевременно обрывались из-за этого ограничения, что и устанавливается проверкой K. При необходимости увеличения T_L весь механизм перебора после процедуры L запускается вновь. Если же резерв времени есть и длина вариантов не ограничивалась, то увеличивается предельный горизонт (процедура M) и все надо считать сначала. Наконец, запаса времени нет, тогда запускается процедура N, и после выбора хода программа кончает работу.

ПОСЛЕСЛОВИЕ

Высказанные в книге общие соображения о системах управления, цели игры, требованиях к алгоритму, искусственном интеллекте, несомненно, нуждаются в проверке и дискуссии. Та же часть книги, где излагается основа алгоритма шахматной игры, алгоритма, который сейчас находится в стадии практической реализации, особо актуальна.

Известно, что алгоритмизация задач перспективного планирования (неточных задач) в наши дни не нашла еще удачных решений. Если шахматная программа, выполненная на основе предлагаемого алгоритма, будет достаточно сильно играть в шахматы, например, как мастер, это будет важным шагом на пути создания искусственного интеллекта.

В приложении 1 Б. Штильман рассказывает о способе получения пучков траекторий и соответствующей ему программе. Получение пучков — основная повторяющаяся операция алгоритма, ее удалось стандартизировать довольно просто.

В приложении 2 А. Юдин знакомит читателя с принципами формирования дебютной библиотеки ЭВМ и программой пользования этой библиотекой. Несмотря на скромные (с точки зрения шахматного мастера) размеры заложенной информации, можно надеяться, что алгоритм поиска хода поможет ЭВМ принимать решения, когда возможности библиотеки будут использованы.

Тот вариант алгоритма, который изложен в данной работе, определился в результате длительной дискуссии. Понятия зоны, МО, избыточного времени (инерционности), неполного перебора (для МО) и другие были из-

вестны читателям книги «Алгоритм игры в шахматы» и двух препринтов Научного Совета по комплексной проблеме «Кибернетика» АН СССР «Блок-схема алгоритма игры в шахматы» (1972 г.) и «О кибернетической цели шахматной игры» (1973 г.). Однако, когда работа над программой была развернута, выяснилось множество неточностей и неясностей (этого и следовало ожидать). Прежде всего, надо отметить тот экономный путь формирования зон, который был предложен Б. Штильманом. Параллельное получение перебора, зон, оценки и обрыва варианта представляется вполне рациональным. Далее следует отметить способ выделения независимых зон, в которых происходит независимый перебор ходов, это приводит к резкому сокращению перебора, что и позволяет оптимистически оценивать возможность успешной проверки программы на современных ЭВМ.

Наконец, о позиционной оценке. С ней было связано много сомнений и неудачных попыток найти разумное решение. Помогла здесь (это можно утверждать, если эксперимент приведет к удовлетворительным результатам) моя работа по редактированию «Учебника шахматной игры» Капабланки. Несколько десятилетий я не заглядывал в эту книгу, теперь, после работы над алгоритмом, я изучал ее с другой точки зрения. Меня поразила решительность, с которой Капа отверг общие советы по позиционной игре, принятые во всех руководствах. Капабланка утверждал, что лишь два фактора играют важную роль в шахматах: материал и контроль полей. Это удивительно близко той цели игры, которая была провозглашена автором этих строк еще десять лет назад. Поэтому и нетрудно было формализовать цель позиционной игры по Капабланке как «контроль полей»!

Автор полагает, что ему удалось убедить читателя в том, что при решении неточных задач успех определяется качеством цели игры (оценочной функции). Когда точная задача решается точно, все просто. Например, если бы в шахматах можно было составить полное, несокращенное дерево перебора и с помощью точной цели игры (мата) определить оптимальный вариант (оптимальные варианты), то не было бы никаких проблем.

Полное дерево перебора состоит как из оптимальных вариантов (их мало), так и из неоптимальных (их много). Когда дерево перебора становится сокращенным (обычный практический случай), точная цель игры уже

бесполезна, появляется неточная, паллиативная цель игры. Примем, что найдена удачная цель игры, тогда из тех вариантов, которые были включены в сокращенное дерево перебора, можно будет найти удачные (если они попали в дерево перебора). Если цель игры фальшива, то отобранные варианты всегда будут слабыми.

Поскольку дерево перебора сокращено, то сокращено и количество вариантов, включенных в это дерево. А поскольку здесь нет всех возможных вариантов, как это имеет место при полном дереве перебора, то выгодно, чтобы сокращенное дерево перебора содержало возможно меньше слабых вариантов, что позволяет увеличить предельную длину вариантов; тогда, несомненно, увеличивается шанс найти хороший вариант. Как было показано, удачная цель игры способствует увеличению количества разумных вариантов и снижает процент слабых. Таким образом, хорошая цель игры позволяет не только отобрать удачный вариант из числа имеющихся в данном дереве, но и улучшить качество самого дерева.

Книга выходит в свет обычно значительно позже того, как рукопись сдается в издательство; хотелось бы высказать надежду, что, когда книга выйдет в свет, ЭВМ еще не будет играть в шахматы с большой силой...

ПРИЛОЖЕНИЕ 1

Формирование множества пучков траекторий

Б. М. Штильман

Шахматы, как известно, представляют собой конечную игру, поэтому можно дать исчерпывающее описание шахмат с помощью дерева игры. Если бы дерево не было столь катастрофически большим, то лучший ход легко определялся бы с помощью минимаксной процедуры на всем дереве игры.

Задача состоит в том, чтобы в полном дереве игры выделить для анализа такое поддерево, которое бы удовлетворяло трем требованиям.

1. В поддереве должны содержаться варианты с хорошим ходом в данной позиции.

2. Поддерево должно быть достаточно маленьким (по числу включенных в него ходов), чтобы устройство по переработке информации могло совершить все необходимые операции за приемлемое время.

3. Поддерево должно быть настолько узким, чтобы при фиксированном числе содержащихся в нем ходов варианты были достаточно длинными и устройство по переработке информации могло получить их разумные оценки и на основании этих оценок выбрать хороший ход.

Созданные к настоящему времени «технологические» программы полного перебора на фиксированную глубину, очевидно, удовлетворяют первому и второму требованиям. Третьему требованию они не удовлетворяют, так как не имеют разумного критерия обрыва вариантов, а также не могут строить глубокого дерева из-за экспоненциального роста числа вариантов. Современные программы сокращенного перебора удовлетворяют лишь требованию 2.

Основой обсуждаемой программы, позволяющей выполнить все требования, является математическое отображение (МО) — динамическая модель шахматной игры. Пусть A — множество всех допустимых правилами ходов в данной позиции; B — такое подмножество A, в котором всегда содержится хороший ход m в данной позиции: $m \in B \subset A$. Задача любого алгоритма неполного перебора ходов состоит в правильном выборе этого подмножества B, т. е. B действительно должно содержать хороший ход и быть небольшим. В нашей программе B — это совокупность некоторых ходов по траекториям МО.

Остановимся на вопросе о том, как создается МО и какую роль играет при этом множество пучков траекторий, а затем перейдем к описанию конкретной реализации этого множества. В исходной позиции рассмотрим пучки траекторий нападения фигур одного цвета на фигуры другого цвета, по длине не превышающие горизонт.

Полученная совокупность пучков и будет исходным множеством пучков в данной позиции С этим багажом мы и начинаем перебор ходов. Фигуры в процессе перебора меняют свои места, начинают «видеть» в пределах горизонта другие фигуры, поэтому появляются

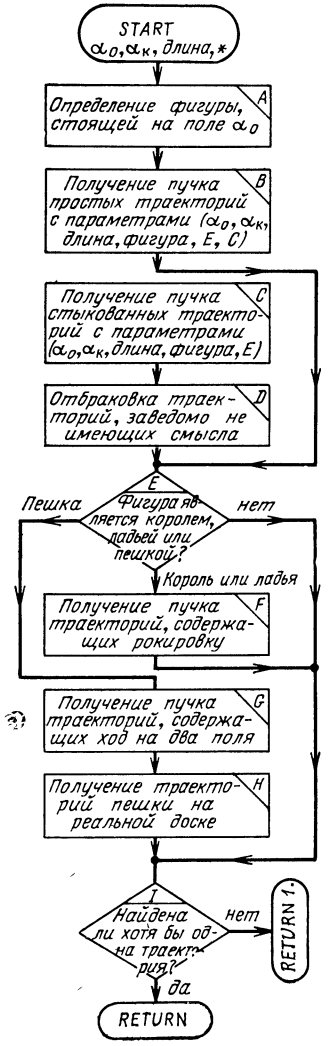

Рис. 8. Блок-схема подпрограммы получения пучка траекторий.

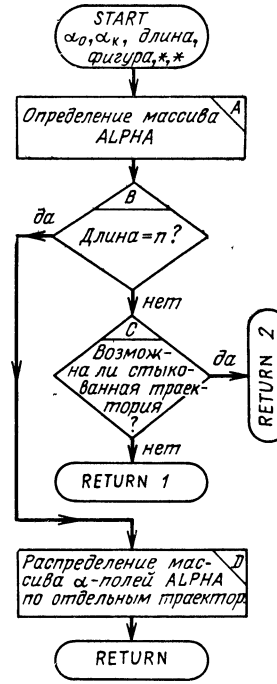

Рис. 9. Блок-схема подпрограммы получения пучка простых траекторий.

новые пучки траекторий — формируются зоны и тем самым создается МО.

Таким образом, перебор ходов здесь используется для сбора информации о структуре зон, которая и позволит найти решение. Если в существующих программах неоптимальные варианты при переборе не дают полезной информации, то в данном алгоритме они используются при формировании зон. Итак, с одной стороны, пере-

бор происходит по траекториям МО, с другой — МО формируется в процессе перебора.

Рассматриваемый здесь подход к алгоритмизации шахмат, в сущности, нигде не использует то, что эта игра является конечной. Методика построения поддерева игры для анализа никак не опирается на существование полного дерева игры. Принципы данного алгоритма могут быть использованы и в играх, где конечного дерева перебора не существует.

Перейдем к описанию программы вычисления множества пучков. Прежде всего, опишем алгоритм получения пучка траекторий с α_0-поля на $\alpha_к$-поле. Для того чтобы воспользоваться алгоритмом, нужно задать поля α_0 и $\alpha_к$, фигуру, траектории которой мы ищем, а также длину пучка, т. е. число передвижений фигуры по траекториям этого пучка на свободной доске. Указанные четыре параметра однозначно определяют пучок траекторий. Однако он может состоять из кратчайших или некратчайших траекторий на свободной от фигур доске. От этого зависит процедура его вычисления. Программа сама определяет, какой же это пучок, и выбирает процедуру его вычисления. Блок-схема этой программы показана на рис. 8.

Остановимся подробнее на подпрограмме получения пучка простых траекторий (рис. 9). На схеме в списке параметров указаны две звездочки, означающие возможность нестандартных выходов из этой подпрограммы RETURN 1 и 2 по адресам, которые будут стоять на месте этих звездочек при обращении к этой подпрограмме.

В процедуре A происходит операция определения массива α-полей пучка траекторий (ALPHA). Это делается следующим образом: массив 15×15, принадлежащий фигуре, пучок которой определяется, совмещается с массивом 8×8 таким образом, что α_0-поле массива 8×8 совмещается с центральным полем массива 15×15. Результат этой операции обозначим $(8\times8\alpha_0)$.

По сути дела происходит передача информации, записанной в массиве 15×15 — TAB15, массиву 8×8 — TAB8 по формуле

$$\text{TAB}\,8(I,\ J)=\text{TAB}\,15(X,\ Y,\ \text{FIGURE}),$$

где $X=8-X_0+I$, $Y=8-Y_0+J$, причем $(X_0,\ Y_0)$ — координаты α_0-поля. Аналогичная операция совершается для $\alpha_к$-поля.

В результате получаются два заполненных числами массива 8×8. Складываем их как векторы, получаем новый массив 8×8. В нем определяем поля, в которых записаны числа, равные длине пучка. Список линейных координат этих полей и образует массив ALPHA.

В процедурах C и B происходит сравнение числа n, стоящего на $\alpha_к$-поле массива $(8\times8\alpha_0)$, с длиной пучка A. Если $A<n$ или $2n_L<A$, то пучок не существует (n_L — предельное число, содержащееся в массиве 15×15 для данной фигуры) и происходит выход из подпрограммы и передача управления по адресу, указанному на месте первой слева звездочки. Если же $n<A\leqslant2n_L$, то возможна стыкованная траектория и снова выход из подпрограммы.

Рассмотрим, что же происходит, если длина пучка равна n. В ALPHA содержатся координаты всех α-полей траекторий пучка. Остается выяснить, как они распределяются по отдельным траекториям, т. е. получить траектории в явном виде (процедура D).

Опишем эту процедуру по индукции. Пусть к данному моменту построено N траекторий пучка, но они вычислены не до конечного

$\alpha_к$-поля, а лишь до α_{i-1}-поля. Покажем, как вычисляются α_i-поля этих траекторий и какие новые траектории при этом появляются.

Сначала строим массив $STEP_i$, состоящий из координат полей массива ($8\times8\alpha_0$), в которых записано число i. $STEP_i$ — это массив полей, на которые наша фигура может попасть с α_0-поля в i передвижений, двигаясь кратчайшими путями. Далее определяем, какие из этих полей содержатся в нашем пучке, т. е. в массиве ALPHA. В результате из общих элементов массивов $STEP_i$ и ALPHA формируется массив RESULT. Это проиллюстрировано на рис. 10. Итак, RESULT — список координат всех α_i-полей траекторий пучка.

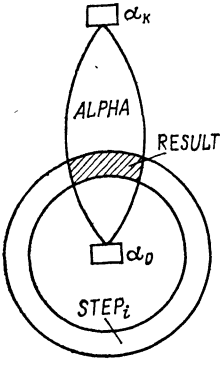

Рис. 10. Определение массива RESULT.

Предположим, что у нас есть список координат всех α_{i-1}-полей траекторий пучка. Назовем его ESULT. Пусть α — элемент массива ESULT. Тогда определим массив $STEP_1(\alpha)$ координат полей,

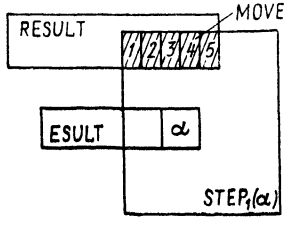

Рис. 11. Определение массива MOVE.

на которые наша фигура может попасть в одно передвижение с поля α. Он определяется аналогично массиву $STEP_i$. Остается выяснить, какие из элементов массива $STEP_1(\alpha)$ содержатся в массиве RESULT. Тем самым мы найдем α_i-поля, на которые можно попасть в одно передвижение с поля α. Координаты этих полей образуют массив MOVE (рис. 11).

Итак, подготовлен материал для наращивания уже имеющихся траекторий и формирования новых. Просматриваем все N уже имеющихся траекторий и ищем среди них такую, в которой $\alpha_{i-1}=\alpha$. Наращиваем эту траекторию α_i=MOVE (1). Формируем новые траектории. Их число равно количеству оставшихся элементов массива MOVE. Новые траектории до α_i-поля совпадают со старой, α_i-поля в них — элементы массива MOVE (рис. 12). Эта операция повторяется для каждой старой траектории, в которой $\alpha_{i-1}=\alpha$.

Описанная процедура выполнялась с элементом α массива ESULT. Теперь переходим к следующему элементу этого массива и для него повторяем процедуру. Исчерпав все элементы массива ESULT, пересылаем массив RESULT в массив ESULT. Таким образом, сделанное выше предположение о существовании массива ESULT обосновано.

N присваиваем число, равное общему количеству построенных к данному моменту траекторий. Теперь можно переходить к нахождению α_{i+1}-полей траекторий

Итак, в данном алгоритме вычисление всех траекторий пучка происходит не по очереди (одна траектория за другой), а параллельно.

Описание процедуры D, а вместе с ней и блок-схемы, изображенной на рис. 9, закончено. Все вышесказанное верно для любой фигуры, кроме пешки и ферзя.

Особенность пешки состоит в том, что в процедуре A для получения массива $(8\times 8\alpha_к)$ нужно использовать массив 15×15 для пешки другого цвета. Что же касается ферзя, то это единственная фигура, для которой эта подпрограмма получает не только кратчай-

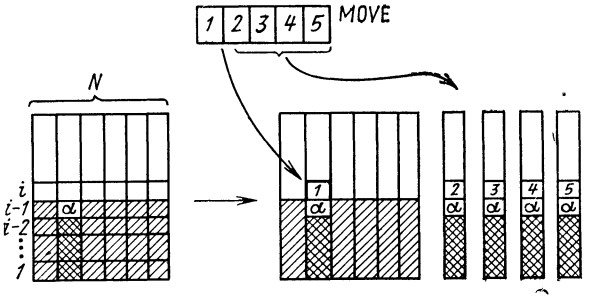

Рис. 12. Вычисление траекторий.

шие траектории, а все траектории в одно и два передвижения. В памяти не хранится специальный массив 15×15 для ферзя. Если подпрограмма занимается поиском пучка траекторий ферзя, то вначале находится пучок, состоящий из траекторий, по которым ферзь передвигается, как слон. Поэтому все процедуры выполняются так, как будто ферзь — слон: Затем все начинается сначала, но находится пучок траекторий, по которым ферзь передвигается, как ладья. После этого строятся еще два пучка траекторий ферзя: «ладья+ +слон» и «слон+ладья» (траектории в два передвижения). Для этого при нахождении массива $(8\times 8\alpha_0)$ используется массив 15×15 для ладьи, а при нахождении массива $(8\times 8\alpha_к)$ — специальный массив, который отличается от массива 15×15 для слона тем, что на пустых полях последнего массива стоят двойки (в дальнейшем будем называть его массив № 7; таким образом, определены семь массивов 15×15: белой пешки, черной пешки, короля, коня, слона, ладьи и массив № 7.). Для пучка «слон+ладья» используется обратная комбинация массивов.

Полученные четыре пучка объединяются вместе и образуют пучок всех траекторий ферзя в два передвижения. Отметим, что если рассматривать отдельно траектории ферзя как ладьи или как слона из этого пучка, то они, конечно, являются кратчайшими для каждой из этих фигур.

Вернемся к описанию блок-схемы, изображенной на рис. 8 Процедура B уже описана. Выясним, что происходит при втором нестандартном выходе из этой подпрограммы (RETURN 2) В списке параметров этой подпрограммы указано, что при таком

выходе надо переходить к процедуре *C* получения пучка стыкованных траекторий. Блок-схема соответствующей подпрограммы изображена на рис 13.

Список параметров здесь аналогичен списку в блок-схеме рис. 9, кроме одного нестандартного выхода, который происходит, если подпрограмма не найдет ни одной траектории. Цель этой подпрограммы — поиск траекторий, составленных из двух кратчайших (на свободной от фигур доске). Следовательно, основная задача — нахождение промежуточного поля, к которому и от которого фигура движется по кратчайшим траекториям. Такие поля назовем полями стыковки. Определение этих полей происходит в процедуре *A* (рис. 13). Она работает так же, как процедура *A* на рис 9, только получающийся в результате список координат полей называется не массивом ALPHA, а списком полей стыковки α_i в массиве 8×8.

Процедуры *B* и *D* состоят в обращении к подпрограмме получения пучка простых траекторий с различными списками входных параметров. В процедуре *B* вычисляется пучок с α_0-поля на α_i-поле стыковки длиной A_1, в процедуре *D* — пучок с α_i-поля на $\alpha_к$-поле длиной A_2, причем A_1 и A_2 подбираются так, чтобы A_1+A_2 равнялась длине искомого пучка стыкованных траекторий. Все нестандартные выходы из этих процедур означают переход к выполнению процедуры *F*, что и отмечено в списках параметров процедур *B* и *D*.

В процедуре *C* пучок, полученный в *B*, переносится в специальный новый массив для того, чтобы использовать освободившееся место для вычисления нового пучка в процедуре *D*. Процедура *C* проиллюстрирована на рис. 14,*а*.

После того, как получены два пучка, относящиеся к одному и тому же полю стыковки, надо произвести собственно стыковку траекторий. Если рассматривать оба пучка как два непересекающихся множества траекторий, то операция стыковки состоит в получении теоретико-множественного произведения этих множеств, т. е. нового множества, состоящего из всех пар, элементами каждой из которых являются две траектории, принадлежащие разным пучкам. Эта операция выполняется процедурой *E* и проиллюстрирована на рис 14,*б*.

Процедуры *B*, *C*, *D* и *E* выполняются для каждого α_i-поля стыковки. Итак, блок-схема рис. 13 полностью описана.

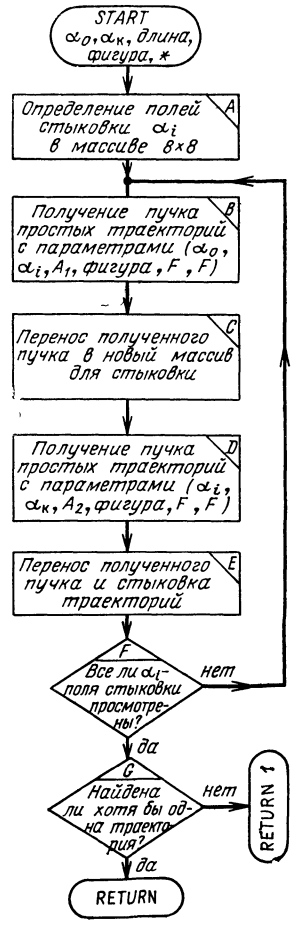

Рис. 13. Блок-схема подпрограммы получения пучка стыкованных траекторий.

Существуют некоторые особенности в работе этой подпрограммы для различных фигур. Нетрудно догадаться, что речь идет о пешке и ферзе. Специфика содержится в единственной операции — определении полей стыковки в процедуре *A*.

Пешка может иметь стыкованные траектории единственного типа: траектории, содержащие превращение пешки в ферзя или другую фигуру. Очевидно, в этом случае полями стыковки могут быть лишь поля первой или последней горизонтали в зависимости от цвета пешки. Для нахождения всех полей стыковки траекторий

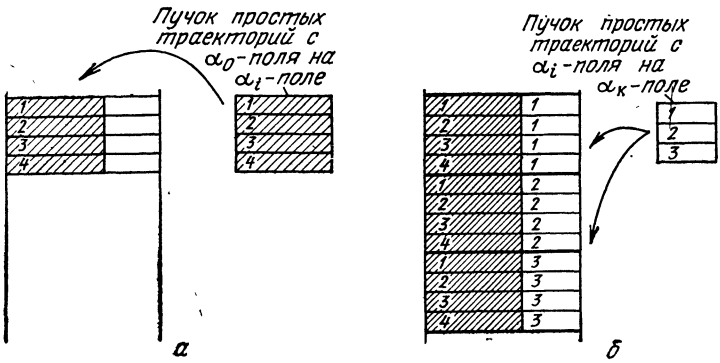

Рис. 14. Перенос пучка простых траекторий в новый массив и стыковка траекторий.

пешки процедура *A* выполняется три раза, каждый раз с новым массивом 15×15, используемым для вычисления ($8\times8\alpha_к$). Речь идет о массивах 15×15 для коня (превращение в коня), ладьи и массиве № 7. Для нахождения массива ($8\times8\alpha_0$) всегда используется один и тот же массив 15×15 для пешки.

Все поля стыковки для ферзя можно определить, выполняя процедуру *A* три раза, каждый раз с новой комбинацией массивов 15×15 при вычислении ($8\times8\alpha_0$) и ($8\times8\alpha_к$):

1) массив 15×15 для ладьи и второй — такой же массив;
2) массив 15×15 № 7 и массив 15×15 для ладьи;
3) массив 15×15 для ладьи и массив 15×15 № 7.

Описание процедуры *C* блок-схемы, изображенной на рис. 8, закончено. Вход в нее может произойти только при нестандартном выходе из *B*. Обратимся к процедуре *D*.

Несмотря на то, что при вычислении пучка стыкованных траекторий мы тщательно определяли поля стыковки, при выполнении стыковки получается много траекторий, заведомо не имеющих смысла. Речь идет о траекториях, имеющих самопересечения, три α-поля подряд, лежащих на одной прямой, и т. п. Выделить и отбросить из пучка эти траектории призвана процедура *D*.

Процедуры *F* и *G* отражают некоторые особенности правил шахматной игры. В процедуре *F* к пучку траекторий ладьи или короля, найденному в предыдущих процедурах, добавляется новый пучок траекторий, первым ходом в которых является рокировка.

В процедуре G вычисляется пучок траекторий пешки, первым ходом в которых является ход на два поля. Отметим, что в этих процедурах используются подпрограммы получения пучков простых и стыкованных траекторий.

Итак, блок-схема алгоритма получения пучка траекторий с α_0-поля на $\alpha_к$-поле описана полностью. Этот алгоритм является основой процедур создания зоны. Программа, работающая по этому алгоритму, получает пучки траекторий, содержащие большое количество информации, особенно, если длина траекторий превышает два передвижения на свободной доске.

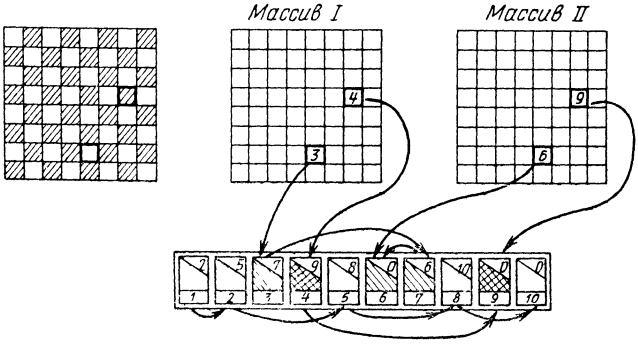

Рис. 15. Размещение множества пучков в оперативной памяти ЭВМ.

Прежде чем воспользоваться этой информацией, следует решить вопрос о ее хранении. Сведения о пучках траекторий нужны на протяжении почти всей работы программы, так как от них зависит формирование зон. Какие же из этих сведений необходимо хранить? Решено хранить в памяти не самый пучок (т. е. списки полей всех траекторий), а лишь информацию о существовании этого пучка и его типе, а именно: номер фигуры, траектории которой содержатся в этом пучке, $\alpha_к$-поле пучка, максимальную длину траекторий пучка, тип пучка (блокады, контроля, деблокады), а также параметры к «застыванию» этого пучка, т. е. параметры, показывающие, что в данный момент перебора движение фигуры по траекториям этого пучка не имеет смысла.

Итак, пучок характеризуется шестью параметрами. Эта информация должна быть «привязана» к полю шахматной доски, на котором стояла фигура, когда этот пучок был впервые найден, т. е. к α_0-полю. По четырем параметрам из перечисленного списка пучок однозначно восстанавливается с помощью описанной выше подпрограммы получения пучка траекторий.

Сведения о пучках можно было бы хранить в виде шестимерного массива, однако учитывая, что каждый из параметров является двузначным десятичным числом (в общем случае), линейная размерность массива составляла бы 10^{12}, а это, конечно, неприемлемо. Поэтому мы храним информацию в виде цепного списка (рис. 15). Память, отводимая для размещения множества пучков, разбивается на «ячейки». На рисунке показаны 10 таких ячеек. Каждая ячейка

предназначена для хранения шести параметров, а также адреса сцепленной с ней ячейки (указан в правом верхнем углу ячейки). Кроме того, отводим память для двух массивов 8×8, нумерация элементов в которых соответствует нумерации полей на шахматной доске.

Информация обо всех пучках с начальным α_0-полем «привязывается» к α_0-полям этих массивов 8×8. Это делается следующим образом: на α_0-поле массива I записывается адрес ячейки, содержащей информацию о первом пучке с начальным α_0-полем, затем все ячейки с информацией о пучках, относящихся к α_0-полю, связываются в цепочку, первым звеном которой является упомянутая выше ячейка. Адрес последнего звена цепочки записывается на α_0-поле массива II. Все ячейки, в которых в данный момент не содержится информации о пучках, также связаны в цепочку.

Подобная система обеспечивает возможность стирания из некоторых ячеек информации, ставшей ненужной после того, как фигура сделает ход в игре, и использования освободившейся памяти для новой информации. Это происходит присоединением освободившихся ячеек к списку пустых.

Если в процессе перебора фигура побывает на α_0-поле, то информация обо всех пучках этой фигуры с началом на α_0-поле будет «привязана» к этому полю. Если при продолжении перебора фигура снова появится на α_0-поле (речь идет о другом варианте перебора), описанная система размещения информации даст возможность восстановить все пучки траекторий этой фигуры, связанные с этим полем и найденные в процессе предшествующего перебора. Это происходит путем последовательного просмотра цепного списка ячеек, связанного с α_0-полем, поиска нужных ячеек, считывания из них параметров пучков и применения подпрограммы получения пучков траекторий с этими параметрами.

Итак, закончено описание программ получения и хранения пучков траекторий.

Во всех рассмотренных выше программах реальная шахматная доска, заставленная фигурами, принималась в расчет лишь в том смысле, что на ней мы определяли начальное и конечное поля пучка. Сами же траектории пучка вычислялись и подвергались предварительному анализу на свободной доске. Информация о пучке, направлявшаяся на хранение, также не содержала никаких данных о заставленности траекторий пучка различными фигурами. Выход на реальную доску осуществляется лишь в момент считывания информации о пучке из памяти и расшифровки этой информации путем вычисления траекторий пучка. Каждую из этих траекторий подвергаем анализу для выяснения, в какой степени она заставлена своими или чужими фигурами. Этот анализ позволяет вычислить истинную длину траектории на реальной доске и решить вопрос о включении ее в игру. Кроме того, мы выясняем необходимость деблокады своих фигур, блокирующих эту траекторию. Анализ начинается с вычисления списка β-полей траектории. После этого проверяем, на каких β-полях стоят фигуры, превращаем их в α-поля. Отметим, что ни в одной из упомянутых выше процедур мы не вычисляли β-полей траекторий, обходясь лишь списком α-полей.

Таким образом, пучок траекторий с β-полями на реальной доске, несущий громадное количество информации, характеризуется всего лишь шестью параметрами. Это дает возможность гибко использовать ресурсы ЭВМ.

ПРИЛОЖЕНИЕ 2

Библиотека дебютов и алгоритм ее использования

А. Д. Юдин

Нет необходимости доказывать пользу применения подпрограммы — библиотеки дебютов для шахматной программы. Четыре чемпионата среди шахматных программ, проведенных в США в 1971—1974 гг., показали, что такие библиотеки имеются у большинства существующих программ. Это, главным образом, экономит время при разыгрывании начальной стадии партии, а также помогает избежать просмотры («зевки») хотя бы в дебюте, вне зависимости от силы игры программы.

Но еще большие преимущества дает использование дебютной библиотеки при создании шахматной программы по рассматриваемому алгоритму. В результате нескольких дебютных ходов, разыгранных с помощью справочника, фигуры сближаются и начинают лучше «видеть» друг друга и большее количество нападений оказывается в пределах заданного горизонта. В данном случае цель «обучения» программы дебютам соответствует той, о которой писал Х. Р. Капабланка *:

«Среднему (речь идет о живом шахматисте, а не о шахматной программе — А. Ю.) любителю для хороших практических результатов необходимо изучить детально пять или шесть дебютов. В дальнейшем он должен изучить еще и другие дебюты. На дебют он должен смотреть просто как на раннюю стадию партии, где целью является методичная и тщательная мобилизация фигур для создания солидной позиции, которая затем позволит ему разрабатывать планы и замышлять комбинации для серединной стадии игры...»

Рассмотрим основные вопросы, связанные с дебютной библиотекой и подпрограммой ее использования для данной шахматной программы.

Библиотека дебютов состоит из пяти деревьев, соответствующих пяти начальным ходам белых: e2 — e4, d2 — d4, c2 — c4, g2 — g3 и Kg1 — f3. Все дальнейшее описание будет относиться к одному дереву, так как начальный ход белых однозначно определяет «рабочее» дерево.

Установлена фиксированная длина вариантов — 6 ходов (или 12 полуходов). Таким образом, после 6-го хода программы происходит безусловная передача управления программы основного алгоритма (если, конечно, эта передача управления не была реализована ранее, что происходит, когда очередной ответ противника не найден в библиотеке).

Далее все варианты были оценены. Рассмотрим возможные оценки (в соответствии с терминологией, принятой «Шахматным информатором»):

0 — игра равна;
+1 — белые стоят лучше;
—1 — черные стоят лучше.

---

* **Капабланка Х. Р.** Учебник шахматной игры. Л-М, ОГИЗ, 1935.

+2 — белые имеют решающее преимущество;
—2 — черные имеют решающее преимущество.

В соответствии с принятыми оценками была проделана процедура минимакса. Таким образом, каждый ход данного дерева получил оценку.

Кодирование хода происходит следующим образом. Каждый ход записан в виде $\pm rmn$, где $\pm r$ — оценка; m и n — координаты начального и конечного полей положения фигуры при данном ходе, которые могут принимать значения от 1 до 64 (рис. 16). Так, например, ход Кс3 — e4, ведущий к решающему преимуществу белых, будет записан в виде: +21929.

8	57	58	59	60	61	62	63	64
7	49	50	51	52	53	54	55	56
6	41	42	43	44	45	46	47	48
5	33	34	35	36	37	38	39	40
4	25	26	27	28	29	30	31	32
3	17	18	19	20	21	22	23	24
2	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8
	A	B	C	D	E	F	G	H

Рис. 16. Кодирование полей доски.

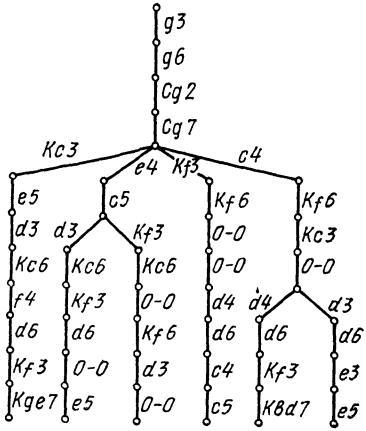

Рис. 17. Дерево вариантов.

Рокировки и взятия на проходе имеют специальные коды.

При записи полей доски в виде массива из 64 элементов ходы-взятия будут записаны как обычные ходы, например ход Кс3 : e4 с оценкой +2 будет записан в виде +21929.

Дерево вариантов записано в виде трех массивов, условно названных LIB, SON и BRAT, имеющих одинаковую длину. Элементы массива LIB — закодированные ходы с оценками. Элементы массивов SON и BRAT — обычные номера (индексы) соответствующих ходов массива LIB или нули.

Рассмотрим пример записи дерева вариантов (рис. 17) в виде трех массивов (рис. 18). Для удобства чтения в массиве LIB записаны незакодированные ходы.

На рис. 19 приведена блок-схема алгоритма пользования библиотекой дебютов. Поясним кратко ее работу.

START передает управление процедуре *A*, которая выясняет цвет сторон. Если программа играет белыми, то процедура *C* при помощи (не показано) процедуры *J* выбирает начальный ход (из набора возможных начальных ходов белых). Далее управление передается (из процедур *B* или *C*) процедуре *D*, за исключением того случая, когда противником, играющим белыми, сделан ход не из

набора e2 — e4, d2 — d4, c2 — c4, g2 — g3, Kg1 — f3, и управление передается программе основного алгоритма.

В процедуре *D* происходит вызов в оперативную память машины библиотечных массивов, соответствующих выбранному дереву, и их подготовка к работе.

Процедуры *E* и *F* находят в библиотеке возможные ответы на последний сделанный (любой стороной) ход и передают управление проверке *G*.

№	LIB	SON	BRAT	№	LIB	SON	BRAT
1	g2—g3	2	0	24	Kg8—f6	25	0
2	g7—g6	3	0	25	d2—d3	26	0
3	Cf1—g2	4	0	26	0—0	0	0
4	Cf8—g7	5	0	27	Kg1—f3	28	35
5	Kb1—c3	6	13	28	Kg8—f6	29	0
6	e7—e5	7	0	29	0—0	30	0
7	d2—d3	8	0	30	0—0	31	0
8	Kb8—c6	9	0	31	d2—d4	32	0
9	f2—f4	10	0	32	d7—d6	33	0
10	d7—d6	11	0	33	c2—c4	34	0
11	Kg1—f3	12	0	34	c7—c5	0	0
12	Kg8—e7	0	0	35	c2—c4	36	0
13	e2—e4	14	27	36	Kg8—f6	37	0
14	c7—c5	15	0	37	Kb1—c3	38	0
15	d2—d3	16	21	38	0—0	39	0
16	Kb8—c6	17	0	39	d2—d4	40	43
17	Kg1—f3	18	0	40	d7—d6	41	0
18	d7—d6	19	0	41	Kg1—f3	42	0
19	0—0	20	0	42	Kb8—d7	0	0
20	e7—e5	0	0	43	d2—d3	44	0
21	Kg1—f3	22	0	44	d7—d6	45	0
22	Kb8—c6	23	0	45	e2—e3	46	0
23	0—0	24	0	46	e7—e5	0	0

Рис. 18. Запись дерева вариантов в виде трех массивов.

При отрицательном результате проверки *E* происходит выход на основной алгоритм.

Если проверкой *G* установлено, что очередь хода за машиной, то с помощью проверки *I* и, если это необходимо, псевдослучайного выбора *J* происходит определение и выдача хода (процедура *L*).

При очереди хода за противником процедура *H* ищет номер этого хода в массиве LIB в наборе индексов, определенном процедурой *F*.

Если ход, сделанный противником, найден в библиотеке, что устанавливается проверкой *K*, то управление передается проверке *M*, в противном случае — мы вне библиотеки дебютов и управление передается программе основного алгоритма.

Наконец, проверка *M* устанавливает, достигнуто ли окончание дебюта. Если да, управление передается программе основного алгоритма, если нет, снова работает проверка *E* и т. д.

Рассмотрим работу процедуры *F*. Допустим, что последним был сделан ход LIB(I), где I — номер этого хода в соответствующем массиве LIB. Необходимо найти возможные (на уровне данной библиотеки) ответы. Они располагаются в массив HOD. Параллельно формируется массив J — номеров, соответствующих ходам массива HOD в массиве LIB. Эти номера определяются по формулам

$$J(k) = \begin{cases} SON(I) \text{ для } k = 1, \\ BRAT\ [J(k-1)] \text{ для } k > 1. \end{cases} \quad (1)$$

Сначала находится SON(I), а затем его «братья» в массиве BRAT. Процесс заканчивается, когда «брат» очередного элемента массива HOD отсутствует (т. е. равен нулю). Искомые ответы находятся по формуле

$$HOD(k) = LIB[J(k)]. \quad (2)$$

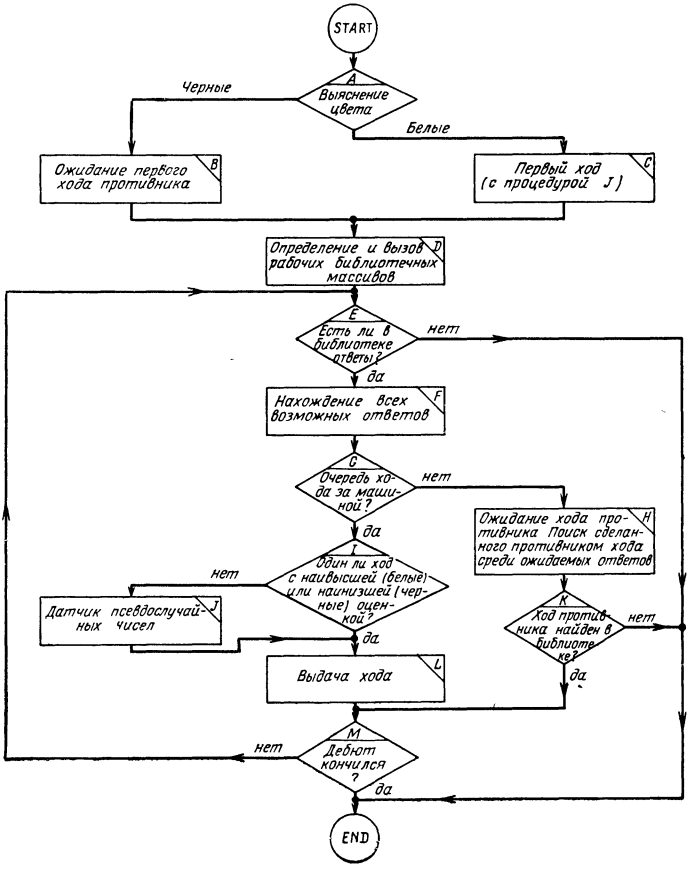

Рис 19. Блок-схема алгоритма использования библиотеки.

При двух или нескольких элементах массива НОД с экстремальными оценками, ход выбирается датчиком равномерно распределенных псевдослучайных чисел (процедура *J*), где в качестве начального числа для псевдослучайного выбора используется текущее время, выраженное в секундах

В настоящее время библиотека дебютов содержит 850 вариантов; объем библиотеки 20 килобайт (при размещении в оперативной памяти машины). Время поиска, выбора и выдачи хода на печать составляет 0,1—0,2 с. Все подпрограммы написаны на языке Фортран IV для машины ICL4-70.

Есть основания считать приведенную структуру дебютной библиотеки достаточно удобной, так как в дальнейшем имеется сравнительно простая возможность (необходимость изменений покажет эксперимент) увеличивать как количество вариантов, так и их длину, корректировать оценки. Такая корректировка и пополнение библиотеки будут происходить в соответствии с тем, как развивается дебютный репертуар шахматиста: библиотека расширяется и видоизменяется на основании практики, сыгранных партий. Включение в дебютную библиотеку вариантов, встретившихся в партиях программы и ранее не включенных в библиотеку, сродни самообучению ЭВМ опять же в соответствии с тем, как происходит самообучение шахматиста любой квалификации.

ПРИЛОЖЕНИЕ 3

Основные определения и обозначения

Дерево игры — граф, каждая вершина которого имеет одну непосредственно предшествующую ей (родительскую) вершину, за исключением выделенной вершины, называемой корнем дерева, которая не имеет предшествующих вершин.

Задача точная — задача, которая может быть решена с помощью формирования полного (несокращенного) дерева перебора и точной цели игры;

неточная — задача, которая может быть решена с помощью формирования неполного (сокращенного) дерева перебора и неточной (паллиативной) цели игры.

Зона игры — совокупность фигур белых и черных, объединенных в поддержке и противодействии атакующей фигуре;

независимая — зона, которая не имеет общих полей и фигур с другими зонами игры;

простая — зона, которая содержит комлевую траекторию (в зоне нападения траекторию нападения) с примыкающими к ней траекториями отрицания;

связанная — зона, которая имеет целью контроль или блокаду поля траектории какой-либо зоны;

сложная — зона, которая образуется несколькими зонами, имеющими общие фигуры и поля.

Избыточное время — время, затрачиваемое на передвижение фигур, не влияющих на результат минимакса.

Конечная игра — игра, когда у каждого игрока имеется только конечное число стратегий, т. е. правил, определяющих выбор варианта действий при каждом ходе этого игрока в зависимости от ситуации, сложившейся в процессе игры.

Математическое отображение позиции (МО) — совокупность зон игры, соответствующих позиции фигур в данный момент игры (перебора).

Поле

α — поле траектории, на котором фигура останавливается при своем движении по траектории;

α_0 — начальное поле траектории;

$\alpha_к$ — конечное поле траектории;

β — поле траектории, которое фигура проходит без остановки.

Псевдоперебор — серия фиктивных ходов, имеющих целью определение новых фигур, участвующих в игре.

Пучок траекторий с α_0-поля на $\alpha_к$-поле данной длины и для данной фигуры — совокупность всех траекторий этой фигуры с α_0-поля на $\alpha_к$-поле, причем таких, что число передвижений фигуры по ним на свободной доске не превышает длины пучка; пучок отступления или деблокады — это совокупность всех траекторий данной фигуры в одно передвижение с поля α_0.

Система управления

полная — система, выполняющая функции получения информации от внешнего мира, переработки информации (с подпрограммой самообучения) и исполнения решения;

частичная — система, у которой отсутствует функция переработки информации или она является неполной, т. е. отсутствует подпрограмма самообучения;

Траектория

вилочная — траектория, часть которой является общей для нескольких траекторий;

кратчайшая данной фигуры с α_0-поля на $\alpha_к$-поле — траектория, которая имеет наименьшее число передвижений на свободной доске;

стыкованная — траектория, составленная из двух кратчайших.

Фигура

(+) — фигура (в простой зоне) одного цвета с фигурой, расположенной на α_0-поле комлевой траектории;

(—) — фигура (в простой зоне) другого цвета нежели фигуры (+);

α_0 — фигура, расположенная на α_0-поле комлевой траектории;

$\alpha_к$ — фигура, расположенная на $\alpha_к$-поле траектории нападения.

Цель игры — оценочная функция, с помощью которой проводится минимакс при отборе оптимального варианта дерева перебора.

Цепной список — список, в котором отдельные члены списка располагаются произвольно в машинной памяти и связываются между собой так называемыми адресами связи. Адрес связи помещается совместно с данным членом списка и указывает положение следующего члена списка.

A и B — числа, определяемые экспериментально, с которыми сравнивается число K для определения возможности позиционной жертвы.

H_L — предельный горизонт нападения, измеряемый в полуходах.

H_x — переменный горизонт контроля полей комлевой траектории, измеряемый в полуходах; значение его зависит от количества

α-полей между атакующей фигурой и контролируемым полем.

K — позиционная оценка.

K_b — число полей, контролируемых черными.

K_w — число полей, контролируемых белыми.

$m_к$ — стоимость фигуры, расположенной на $\alpha_к$-поле траектории нападения в простой зоне.

m_b — стоимость черной фигуры.

m_w — стоимость белой фигуры.

n — число на поле в массиве 15×15, равное числу передвижений фигуры с центрального поля массива до данного поля по кратчайшей траектории.

Δm — результат размена.

$\Sigma m_к$ — сумма стоимостей фигур, расположенных на $\alpha_к$-полях в сложной зоне

Содержание

Ботвинник М. М.

Б 86 О кибернетической цели игры. М., «Сов. радио», 1975.

88 с. с ил.

Рассматриваются общие вопросы работы систем управления, их классификация, требования, которым должен отвечать алгоритм работы этих систем. Отмечается особое значение кибернетической цели игры для составления алгоритма и эффективности работы систем управления. Приводится алгоритм игры в шахматы как пример, поясняющий общие вопросы теории. М. М. Ботвинник — доктор технических наук, экс-чемпион мира по шахматам. Последнее время активно работает в области машинной игры в шахматы.

Книга рассчитана на широкий круг читателей.

Б $\frac{30501\text{-}034}{046(01)\text{-}75}$ 91-74 6ФО.1

МИХАИЛ МОИСЕЕВИЧ БОТВИННИК

О КИБЕРНЕТИЧЕСКОЙ ЦЕЛИ ИГРЫ

Под редакцией Н. А. КРИНИЦКОГО

Редактор *Н. Г. Давыдова*
Художественный редактор *В. Т. Сидоренко*
Технический редактор *З. Н. Ратникова*
Корректоры: *Т. М. Толмачева, Г. М. Денисова*

Сдано в набор 26/XII 1974 г. Подписано в печать 14/III 1975 г.
Т-05739 Формат 84×108/32 Бумага машиномелованная
Объем 4,62 усл. печ. л. 5,170 уч.-изд. л.
Тираж 57 000 экз. Зак. 38 Цена 18 коп.
Издательство «Советское радио», Москва, Главпочтамт, а/я 693

Московская типография № 10 «Союзполиграфпрома»
при Государственном Комитете Совета Министров СССР
по делам издательств, полиграфии и книжной торговли.
Москва, М-114, Шлюзовая наб., 10.

18 к.