Энгельсский технологический институт (филиал) федерального государственного
бюджетного образовательного учреждения
высшего образования
«Саратовский государственный технический университет имени Гагарина Ю.А.»

Задачи и методы решения систем искусственного интеллекта

Методическое указание к практическим работам по дисциплине
«Системы искусственного интеллекта для студентов направлений

09.03.01 «Информатика и вычислительная техника»
09.03.04 «Программная инженерия»
15.03.02 «Технологические машины и оборудование»
15.03.05 «Конструкторско-технологическое обеспечение машиностроительных производств»
18.03.01 «Химическая технология»
21.03.01 «Нефтегазовое дело»
29.03.05 «Конструирование изделий легкой промышленности»

очной и заочной форм обучения

Энгельс 2026

ЗАДАЧИ И МЕТОДЫ ИХ РЕШЕНИЯ

1 Задачи систем искусственного интеллекта

Задачи систем искусственного интеллекта охватывают самые разные предметные области, среди которых лидируют бизнес, производство, медицина, проектирование и системы управления. Все задачи можно классифицировать по следующим общим основаниям [1–3, 7–9].

1) Задачи анализа и синтеза. В задаче анализа задана модель сущности и требуется определить неизвестные характеристики модели. В задаче синтеза задаются условия, которым должны удовлетворять характеристики «неизвестной» модели сущности, и требуется построить модель этой сущности.
2) Статические и динамические. В статических задачах явно не учитывают фактор времени и/или не изменяют знания об окружающем мире в процессе своих решений.
3) Использование общих утверждений для представления знаний, не содержащих явных ссылок на конкретные сущности, которые необходимо определить.
4) Использование частных ссылок, содержащие ссылки на конкретные сущности (объекты).

По типу решаемой задачи различают следующие задачи:

- интерпретация: процесс определения смысла данных (построение описаний по наблюдаемым данным);
- диагностика: процесс соотнесения объекта с некоторым классом объектов и/или обнаружение неисправностей в системе (отклонение параметров системы от нормативных в технике и в живых организмах);
- мониторинг: непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы;

- прогнозирование: построение планов действий объектов, будущих событий на основе моделей прошлого и настоящего. В прогнозирующих системах часто используют динамические модели, в которых значения параметров «подгоняются» под заданную ситуацию. Выводимые из этих моделей следствия составляют основу для прогнозов с вероятностными оценками;
- планирование: конструирование плана действий объектов способных выполнять некоторые функции, т. е. программы действий. Оно основано на моделях поведения реальных объектов, которые позволяют проводить логический вывод последствий планируемой деятельности;
- проектирование: разработка ранее не существовавшего объекта и подготовка спецификаций на создание объектов с заранее определенными свойствами. Степень новизны может быть разной и определяется видом знаний и методами их обработки;
- обучение: диагностика, интерпретация, планирование, проектирование. Системы обучения выполняют такие функции, как диагностика ошибок, подсказывание правильных решений, аккумулирование знаний о гипотетическом «ученике» и его характерных ошибках, диагностирование слабости в познаниях обучаемых и нахождение соответствующих средств для их ликвидации. Системы обучения способны планировать акт общения с учеником;
- управление: интерпретация, прогноз, планирование, моделирование, оптимизация выработанных решений, мониторинг, т. е. функция системы, поддерживающая определенный режим ее функционирования или управления поведением сложной системы в соответствии с заданными спецификациями;
- отладка, ремонт: выработка рекомендаций по устранению неисправностей;
- поддержка принятия решений — совокупность процедур, обеспечивающих ЛПР необходимой информацией и рекомендациями для процесса принятия решений (выбор и/или, генерация альтернатив).

Задачи интерпретации данных, диагностики, поддержки принятия решений относятся к задачам анализа, задачи проектирования, планирования и управления — к задачам синтеза. К комбинированному типу задач относятся обучение, мониторинг и прогнозирование.

Термин «решение задач» (*problem solving*) употребляется в искусственном интеллекте в ограниченном смысле. Речь идет о хорошо определенных задачах, решаемых на основе поисковых алгоритмов.

Задача считается хорошо определенной, если для нее имеется возможность задать пространство возможных решений (состояний), а также способ просмотра этого пространства с целью поиска конечного (целевого) состояния, соответствующего решаемой задаче. Поиск конечного состояния задачи заключается в применении к каждому состоянию алгоритмической процедуры с целью проверки, не является ли это состояние решением задачи. Данная процедура продолжается до тех пор, пока не будет найдено решение.

Примерами хорошо определенных задач являются: доказательство теорем, поиск маршрута на графе, планирование робота в среде с препятствиями и т. д.

Человек обычно не решает задачу в той форме, в которой она изначально формулируется. Он стремится представить задачу таким образом, чтобы ему удобно было ее решать. Для этого он выполняет преобразование исходного представления задачи с целью сокращения пространства, в котором необходимо выполнять поиск решения задачи. Этап выбора подходящей формы представления задачи настолько обыден, что мы часто не осознаем его важности. В то же время форма или способ представления задачи в значительной мере определяет успех ее решения. При выборе способа представления задачи обычно учитывают два обстоятельства: представление задачи должно достаточно точно моделировать реальность; способы представления должны быть такими, чтобы решателю задач было удобно с ним работать.

Поскольку под решателем задач в искусственном интеллекте понимается компьютер, то мы и рассмотрим способы представления задач, удобные для их решения на ЭВМ. К ним относятся следующие наиболее часто используемые способы (формы) [4–6]:

- представление задач в пространстве состояний;
- представление, сводящее задачу к подзадачам;
- представление задач в виде теорем.

Данные представления и соответствующие универсальные методы поиска решений разрабатывались преимущественно на начальных этапах развития искусственного интеллекта. Позже было замечено, что для решения многих практических задач одних универсальных стратегий недостаточно. Необходим также большой объем знаний и наличие практического опыта. Исследование в области ИИ сосредоточились на представлении и приобретении знаний. Однако это не снизило значимости разработанных стратегий поиска решений, так как они представляют некоторые общие схемы управления механизмом вывода систем, основанных на знаниях. Рассматриваемые ниже способы представления задач и методы поиска их решений играют важную роль во многих системах ИИ, включая экспертные системы, понимание естественного языка, доказательство теорем и обучение.

2 Общие способы решения задач

Процесс решения задачи, как правило, включает два этапа: представление задачи и поиск (перебор). Успех решения задачи в значительной мере определяется формой ее представления. Формы представления задачи могут быть различными и зависят как от природы самой задачи, так и от ее решателя.

Пример

Например, при вычислении интеграла человек стремится путем преобразований представить его так, чтобы воспользоваться его табличными интегралами.

Выбирая кратчайший маршрут движения, человек может воспользоваться схемным представлением возможных путей перемещения и оценками расстояний между промежуточными пунктами.

Характер человеческого мышления таков, что этап и форма представления задачи, которую он использует, не всегда им осознаются. Так, шахматист не может четко объяснить форму представления шахматной ситуации, позволяющую ему не перебирать все возможные варианты продолжения при поиске очередного хода. Поэтому этап представления задачи часто выпадает из поля зрения человека. Тем не менее важность этого этапа осознается сразу же при попытке построить программно реализуемый алгоритм решения задачи.

Поиск формы представления задачи, удобный для ее машинного решения, является трудно формализуемым творческим процессом. Можно выделить следующие употребительные формы: представление в пространстве состояний, представление путем сведения задачи к подзадачам, представление в виде теоремы, комбинированное представление.

Полное представление задачи в пространстве состояний включает описание всех состояний или только начальных, задание операторов, отображающих одни состояния в другие, и задание целевого состояния.

Возможны различные формы описания состояний задачи. В частности, могут быть использованы строки, векторы, матрицы и графы. Выбирая ту или иную форму описания состояний, следует позаботиться о том, чтобы применение оператора, преобразующего одно состояние в другое, оказалось достаточно простым.

Операторы могут быть заданы с помощью таблицы, связывающей каждое входное состояние с некоторым выходным. Для больших задач такое задание оператора практически затруднительно. При описании состояния в форме строки (вектора) удобно задать оператор в виде правила переписывания.

Процедура поиска решения в пространстве состояний состоит в том, чтобы найти последовательность операторов, которая преобразует начальное состояние в целевое. Решением задачи будет указанная последовательность операторов.

Деревом называется ориентированный граф, в каждую вершину которого входит только одна дуга, за исключением одной вершины, называемой корнем дерева.

Таким образом, в дереве каждая вершина, за исключением корня, является концом ровно одной дуги и началом одной или нескольких дуг.

Вершины V_i порождаются вершиной V. V — родительская вершина, а V_i — дочерние вершины.

Примем, что корень находится на 0 уровне. Вершины, порожденные корнем, — 1 уровень и т. д.

Существуют два типа структур взаимосвязи подзадач: И-структуры и И-ИЛИ-структуры. В структурах типа И для решения основной задачи требуется решить все подзадачи. В структурах И-ИЛИ подзадачи разбиваются на группы, внутри которых они связаны отношением И, а между группами — отношением ИЛИ.

В этом случае для решения исходной задачи достаточно решить все подзадачи только какой-либо одной группы.

Для описания представления сведения задач к подзадачам можно использовать граф, называемый графом редукции задачи (рис. 1). При этом вершинам будут соответствовать задачи, а дугам — операторы редукции задач. Корню соответствует

исходная задача, вершинам 1-го уровня — задачи, порожденные исходной задачей.

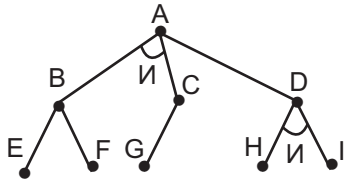


Рис. 1 – Дерево редукции задачи.

Задача *A* может быть решена, если будут решены задачи *B* и *C* или задача *D*. Задача *B* будет решена, если будут решены задачи *E* или *F*. Задача *C* — если решена *G*. Задача *D* — если решены задачи *H* и *I*.

Для указания связности вершин используется специальная кривая. Если имеются связанные вершины (дуги), то обычно, вводя при необходимости дополнительные вершины, дерево редукции задачи преобразуют так, чтобы каждая группа связанных вершин имела отдельную родительскую вершину (рис. 2).

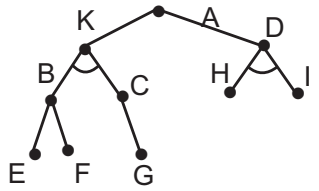


Рис. 2 – Преобразованное дерево редукции.

Будем рассматривать только такие деревья редукции.

Пример

Задача о выборе маршрута, или задача о коммивояжере.

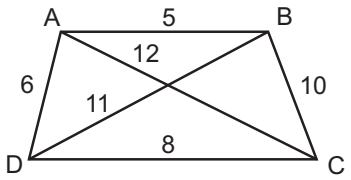


Рис. 3

Транспортный робот должен построить маршрут так, чтобы побывать в каждом из *n* заданных пунктов в точности по разу и возвратиться в исходный пункт. При этом если таких маршрутов несколько, желательно выбрать такой, который имеет минимальную протяженность.

Состояния в этой задаче можно задавать строкой, обозначающей список пунктов, пройденных к текущему моменту.

Операторы — это отображения, соответствующие решению робота направиться из данного пункта в следующий. Целевым состоянием, если не требуется минимальная протяженность маршрута, является любое состояние, описание которого начинается и кончается A и содержит названия всех других пунктов. Например, состояние $ABCDA$.

Представление задачи, сводящее задачу к подзадачам, предусматривает разбиение исходной задачи на множество подзадач, раздельное решение которых дает решение исходной задачи. Каждая из подзадач может, в свою очередь, быть также разбита на подзадачи. Число уровней разбиения теоретически не ограничено. На практике разбиение продолжается до получения на нижнем уровне множества задач (подзадач), способ решения которых известен. Такие задачи условимся называть элементарными.

Введем понятия теории графов. Графом (или геометрическим графом) называется геометрическая конфигурация, состоящая из множества V точек, взаимосвязанных с множеством E непрерывных, самонепересекающихся кривых. Точки из множества V называются вершинами, кривые из множества E — ребрами. Если на всех ребрах задано направление, то граф называют ориентированным графом, а его ребра — дугами.

Если заданы два графа $G1$ и $G2$, причем множества вершин и кривых графа $G2$ являются подмножествами множеств вершин и кривых $G1$, то $G2$ называют подграфом графа $G1$, $G1$ — надграфом графа $G2$.

Структурно граф (дерево) редукции задачи, отличается от графа (дерева) состояния тем, что в нем имеются связанные дуги. Связанные вершины называют И-вершинами, несвязанные — ИЛИ-вершинами, а граф называется графом типа И-ИЛИ.

Вершины, соответствующие элементарным задачам называются заключительными.

Вершины, не имеющие дочерних вершин и не являющиеся заключительными, называются тупиковыми. Тупиковым вершинам соответствуют задачи, которые в рамках данного представления неразрешимы.

Таким образом, заключительные вершины являются разрешимыми, а тупиковые — неразрешимыми.

Вершина, не являющаяся ни заключительной, ни тупиковой, будет разрешимой тогда и только тогда, когда все ее дочерние вершины разрешимы, если они являются связанными, или хотя бы одна из дочерних вершин разрешима, если они являются несвязанными.

Очевидно, задача A является разрешимой в том и только в том случае, если вершины H и I являются заключительными или заключительными являются вершины G и F или G и E.

Граф редукции задачи может быть задан в явном виде (на рис. 2.2). Но чаще он, как и граф состояния, задается в неявном виде посредством описания исходной задачи и операторов редукции.

3 Методы решения задач

Методы решения задач, основанные на сведении их к поиску, зависят от особенностей предметной области, в которой решается задача, и от требований, предъявляемых пользователем к решению. Особенности предметной области:

1) объем пространства, в котором предстоит искать решение;
2) степень изменяемости области во времени и пространстве (статические и динамические области);
3) полнота модели, описывающей область. Если модель не полна, то для описания области используют несколько моделей, дополняющих друг друга;
4) определенность данных о решаемой задаче, степень точности (ошибочности) и полноты (неполноты) данных.

Требования пользователя к результату задачи, решаемой с помощью поиска, можно характеризовать:

1) количеством решений: одно решение, несколько решений, все решения;
2) свойствами результата: ограничения, которым должен удовлетворять полученный результат;
3) и (или) способом его получения.

Существующие методы решения задач, используемые в интеллектуальных системах, можно классифицировать следующим образом [1, 7]:

1) методы поиска в одном пространстве — методы, предназначенные для использования в следующих условиях: области небольшой размерности, полнота модели, точные и полные данные;
2) методы поиска в иерархических пространствах — методы, предназначенные для работы в областях большой размерности;
3) методы поиска при неточных и неполных данных;
4) методы поиска, использующие несколько моделей, предназначенные для работы с областями, для адекватного описания которых одной модели недостаточно.

Предполагается, что перечисленные методы при необходимости должны объединяться для того, чтобы позволить решать задачи, сложность которых возрастает одновременно по нескольким параметрам.

3.1 Поиск решений в одном пространстве

Методы поиска решений в одном пространстве обычно делятся:

1) на поиск в пространстве состояний (рассмотрим подробно);
2) поиск методом редукции;
3) эвристический поиск;
4) поиск методом «генерация-проверка».

Поиск в пространстве состояний

В большинстве СИИ информация, доступная стратегии управления, явно недостаточна для того, чтобы выбрать подходящее правило на каждом шаге процесса, поэтому сами процедуры вывода являются, как правило, поисковыми процедурами. Поскольку понятиям знания о задаче могут соответствовать состояния задачи, а правилам вывода — операторы перехода из одного состояния в другое, то процедура поиска решения называется поиском в пространстве состояний.

Поиск решений в пространстве состояний сводится к определению последовательности операторов, отображающих начальные состояния в целевые. Причем если такая последовательность не одна и задан критерий оптимальности, то поиск сводится к нахождению оптимальной последовательности операторов, обеспечивающих оптимум заданного критерия оптимальности.

Методы поиска решений в пространстве состояний удобно рассмотреть, используя дерево (граф) состояний. На дереве состояний поиск решения сводится к определению пути (оптимального, если задан критерий оптимальности) от корня дерева к целевой вершине, т. е. к вершине, соответствующей целевому состоянию. Поиск решения можно наглядно проиллюстрировать на дереве состояний, когда начальное состояние одно. Поэтому сначала рассмотрим задачи, определяемые тройкой (S_0, F, G), в которой множество S_0 начальных состояний состоит из одного элемента, F — множество операторов, G — множество целевых состояний.

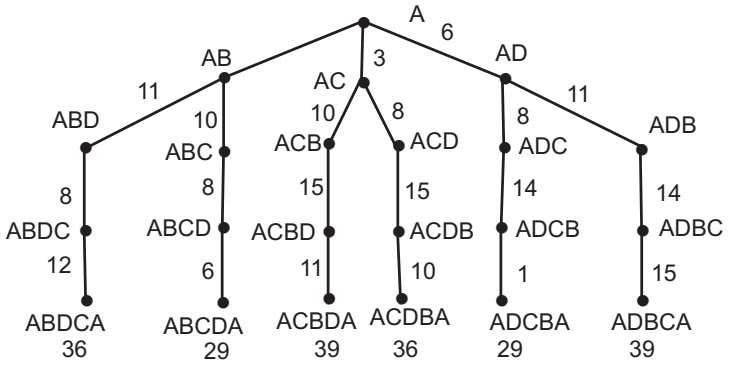


Рис. 4 – Граф состояния задачи.

Для построения дерева состояния следует, используя операторы из F, применимые к корню дерева (начальное состояние), построить вершины 1-го уровня. Затем, используя операторы из F, применимые к вершинам 1-го уровня, построить вершины 2-го уровня и т. д. Процесс применения операторов к какой-либо вершине дерева для построения всех ее дочерних вершин называется раскрытием вершин. Поэтому операторы преобразования мира (операторы из F) интерпретируются как правила раскрытия вершин. Применение правил раскрытия к начальной вершине порождает совокупность дочерних вершин. Каждая из них соответствует некоторому состоянию мира, в которое он может перейти из начального состояния. Дуги, связывающие начальную вершину с дочерней, идентифицируют как соответствующие операторы преобразования. Для всех дочерних вершин производится проверка, не являются ли они целевыми, т. е. не соответствуют ли целевым состояниям. Если целевая вершина не обнаружена, то выполняется следующий

этап порождения вершин путем применения правил раскрытия к каждой вершине, порожденной на предыдущем этапе, и т. д. Процедура продолжается до обнаружения целевой вершины.

Рассмотрим процедуру построения графа состояний на примере выбора маршрута транспортным роботом, который должен, выйдя из склада (пункт *A*), обойти обрабатывающие центры (пункты *B*, *C*, *D*) и вернуться в склад. Запрещается, чтобы робот побывал в каком-либо из обрабатывающих центров более одного раза. Схема маршрутов и расстояние между пунктами представим на рис. 4.

Условимся состояние в этой задаче обозначать словом (набором букв), образованным из букв, соответствующих наименованию пунктов, пройденных к текущему моменту и расположенных в том порядке, в каком были пройдены соответствующие пункты. Тогда очевидно, что начальным будет состояние *A*, а целевыми будут состояния, которые начинаются и оканчиваются словами *A* и содержат остальные буквы по одному разу.

Операторы (или правила раскрытия) в данном примере соответствуют выбору того или иного маршрута.

На графе состояний этой задачи начальной вершине (корню дерева) соответствует состояние *A*. Корень дерева порождает три дочерние вершины, соответствующие состояниям *AB*, *AC*, *AD*. Каждая из вершин, порожденных корнем, порождает по две вершины и каждая из вершин 2 и 3 уровней — по одной. На дугах указаны расстояния, которые проходит робот при переходе из одного состояния в другое.

Поиск решения имеет итеративный характер, причем число итераций и вершин, раскрытых до нахождения целевой вершины, существенно зависит от порядка (последовательности), в котором раскрывались вершины. Порядок раскрытия вершин принято называть стратегией поиска.

Можно выделить два основных типа стратегий поиска: «слепой» перебор и упорядоченный перебор вершин-кандидатов на раскрытие. «Слепой» перебор характеризуется тем, что расположение целевых вершин или их близость не влияют на порядок раскрытия. Существует несколько алгоритмов «слепого» перебора. Рассмотрим три наиболее типичных: алгоритм полного перебора, алгоритм равных цен, алгоритм перебора в глубину [4–6].

Алгоритм полного перебора. Вершины раскрываются в том порядке, в котором они были порождены. Первой раскрывается начальная вершина. Проверяется, нет ли среди порожденных вершин целевой. Если есть, то поиск заканчивается. Если нет, то раскрывается первая из порожденных вершин и проверяется наличие целевых вершин. Затем раскрывается вторая из порожденных вершин и т. д.

Для структурированной записи алгоритмов поиска в пространстве состояний введем понятия списков «открытых» и «закрытых» вершин. Список «закрытых» вершин — это список, где размешаются идентификаторы «раскрытых» и идентификатор вершины, которую предстоит раскрыть, в данный момент. Вершины из списка «закрыт», кроме последней, раскрывать нельзя.

Список «открытых» вершин — это список, где размещаются вершины, которые могут и должны быть раскрыты. Стратегии поиска различаются правилами размещения вершин в списке «открыт» и выбора очередной вершины для раскрытия.

В структуризованном виде алгоритм полного перебора можно представить следующим образом:

1) поместить начальную вершину в список «открыт»;
2) если список «открыт» пуст, то подать сигнал о неудаче поиска, в ином случае перейти к следующему шагу;
3) взять первую вершину из списка «открыт» и перенести ее в список «закрыт»; присвоить вершине идентификатор v;
4) раскрыть вершину v. Поместить все дочерние неповторяющиеся (т. е. не встречающиеся в списке «закрыт») вершины в конец списка «открыт» и построить указатели, ведущие от них к вершине v. Если вершина v не имеет дочерних вершин или имеет только повторяющиеся дочерние вершины, то перейти к шагу 2;
5) проверить, не является ли одна из дочерних вершин v целевой. Если является, то выдать решение; в ином случае перейти к шагу 2.

В этом алгоритме предполагается, что начальная вершина (корень) не может быть целевой. Алгоритм, безусловно, позволяет найти оптимальное (минимальное по числу дуг) решение. Например, использование этого алгоритма для решения задачи о выборе маршрута (с минимальным расстоянием) по существу сводится к построению графа состояний. Имея этот граф и сравнивая расстояния различных маршрутов, ведущих к целевым состояниям, можно выбрать оптимальные маршруты. Как следует из графа состояний, таких маршрутов два — $ABCDA$, $ADCBA$.

Алгоритм перебора в глубину. Определим глубину вершины числом, равным номеру ее уровня. При методе перебора в глубину всегда вскрывается та из вершин, которая имеет наибольшую глубину. Так как несколько вершин могут иметь одинаковую наибольшую глубину, предполагается, что имеется правило выбора одной из них. Кроме того, обычно по тем или иным соображениям задается граничная глубина; вершины, имеющие глубину, равную граничной, не раскрываются. Таким образом, метод перебора в глубину можно определить как метод перебора, при котором всегда раскрывается та из вершин, которая имеет наибольшую глубину, меньшую граничной.

Рассмотрим алгоритм перебора в глубину в структуризованном виде:

1) поместить начальную вершину в список «открыт»;
2) если список «открыт» пуст, то выдается сообщение о неудаче, в ином случае перейти к шагу 3;
3) взять первую вершину из списка «открыт» и перенести в список «закрыт». Этой вершине присвоить идентификатор v;
4) если глубина вершины v равна граничной глубине, то перейти к 2, в ином случае к 5;
5) раскрыть вершину v. Поместить все дочерние вершины в начало списка «открыт» и построить указатели, идущие от них к вершине v. Если v не имеет дочерних вершин, то перейти к шагу 2;
6) если одна из этих вершин целевая, то на выход выдать решение, получаемое путем просмотра назад в соответствии с указателями, в ином случае перейти к шагу 2.

Приведем один из возможных графов решения задачи выбора маршрута с помощью алгоритма перебора в глубину. При применении этого алгоритма решение

зависит от стратегии упорядочения вершин в списке «открыт». В приведенном решении найден неоптимальный маршрут (рис. 5).

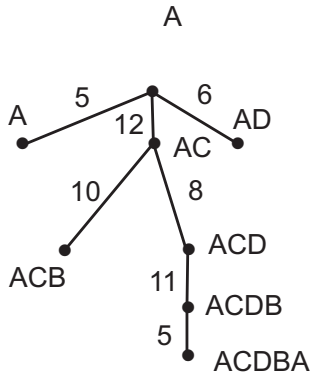


Рис. 5 – Граф решения задачи о выборе маршрута при использовании перебора в глубину.

Во всех рассмотренных алгоритмах перебора предполагается, что начальная вершина только одна. Если начальных вершин несколько, то эти алгоритмы изменяются на шаге 1: на шаге 1 в список «открыт» помещаются все начальные вершины.

Достоинством методов «слепого» перебора является, во-первых, простота алгоритмической реализации, во-вторых, обязательность получения решения, если оно существует. Недостатком этих методов является резкое возрастание числа вершин, которые необходимо раскрыть в процессе поиска решения, т. е. увеличение размерности задачи. Это существенно сужает круг практических задач, которые могут быть решены методами «слепого» перебора.

Алгоритм упорядоченного перебора. Для большинства практических задач удается сформулировать эмпирические правила, позволяющие уменьшить объем перебора. Эти правила используют специфическую информацию о решаемой задаче, формулируемую на основе опыта, интуиции и здравого смысла исследователя. Информацию такого рода часто называют *эвристической*, а основанные на ней алгоритмы — *эвристическими*.

Основная идея эвристических алгоритмов заключается в упорядочении списка открытых вершин в соответствии с некоторой мерой, оценивающей «перспективность» вершины или пути, на котором находится данная вершина. Такую меру называют *оценочной функцией*. Для очередного раскрытия из списка «открыт» выбирается вершина, имеющая минимальное значение оценочной функции. После каждого шага раскрытия производится переупорядочение вершин в списке в соответствии со значениями оценочной функции. Процедуру такого типа называют *алгоритмом упорядоченного перебора*. Существуют различные идеологии построения оценочных функций. Рассмотрим наиболее распространенную [4, 6, 7].

Пусть $g(v)$ — стоимость кратчайшего (оптимального) пути из любой начальной вершины $s_0 \in S_0$ до некоторой вершины v. Стоимость кратчайшего пути из вершины v до ближайшей целевой вершины обозначим $h(v)$.

Функция $f(v) = g(v) + h(v)$, очевидно, выражает стоимость кратчайшего (оптимального) пути из начальной вершины в целевую при условии, что он проходит через вершину v. Введем в рассмотрение следующие оценочные функции: $\hat{g}(v)$ — оценка стоимости кратчайшего пути из начальной вершины в вершину v,

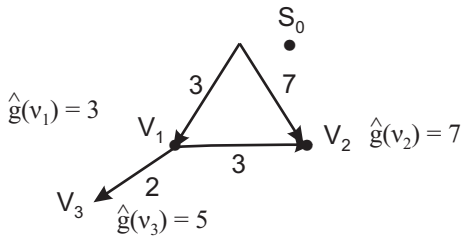
$\hat{h}(v)$ — оценка стоимости кратчайшего пути из вершины v до ближайшей целевой.

Тогда функция $\hat{f}(v) = \hat{g}(v) + \hat{h}(v)$ будет оценочной функцией функции $f(v)$, т. е. она является оценкой стоимости кратчайшего пути из начальной вершины в целевую при условии, что он проходит через вершину v.

В качестве оценочной функции $\hat{g}(v)$ естественно выбрать действительную стоимость пути от начальной вершины к вершине v найденного алгоритмом перебора к данному моменту. Заметим, что если граф состояний не имеет структуру дерева, то значение $\hat{g}(v)$ может меняться в процессе раскрытия вершин. Поясним это свойство на примере.

Пусть на шаге 1 поиска раскрыта вершина S_0 и порождены вершины v_1, и v_2 со значениями стоимостей $\hat{g}(v_1) = 3$, $\hat{g}(v_2) = 7$.

На шаге 2 раскрывается вершина v_1, и порождается вершина v_3 и снова вершина v_2, причем из v_1 в v_2 ведет путь стоимостью $g(v_1) = 3$. Очевидно, стоимость кратчайшего пути из S_0 в v_2 будет $g(v_2) = 6$.



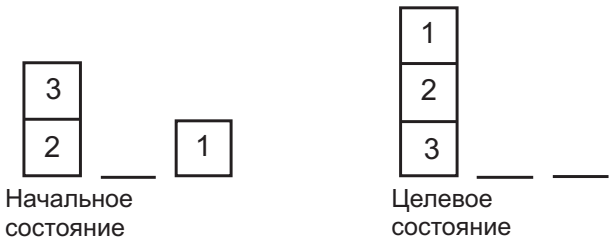
Необходимо заметить, что $\hat{g}(v) \geqslant g(v)$. Это видно и из примера. Эта оценочная функция легко вычисляется в процессе работы алгоритма.

Определение оценочной функции $\hat{h}(v)$ является более сложной задачей. При ее построении опираются на любую эвристическую информацию о решаемой задаче, поэтому $\hat{h}(v)$ называют эвристической функцией. Очевидно, если $\hat{h}(v) = 0$, то алгоритм перебора сводится к алгоритму равных цен, что соответствует полному отсутствию эвристической информации. Не существует каких-либо конструктивных рекомендаций к способам определения этой функции, поэтому рассмотрим ее смысл на примере. Конкретизируем задачу преобразования сцен (задачу о манипуляции предметами) следующим образом.

Пример

Пусть состояние, или, как принято говорить, мир задачи включает некоторое число площадок с расположенными на них кубиками (в реальной задаче вместо кубиков могут рассматриваться детали, из которых необходимо собрать изделие), а также робот манипулятор, перемещающий кубики с одной площадки на другую и устанавливающий их друг на друга. Текущее состояние такого мира можно интерпретировать как текущее расположение кубиков на площадках и положение манипулятора. Совокупность всех возможных состояний образует множество S. Очевидно, способы перемещения кубиков роботом, т. е. переход от одного состояния к другому, должны быть подчинены некоторым требованиям. Так, может быть ограничено число кубиков на конкретной площадке. Для перемещения естественно выбирать кубики, лежащие сверху, и т. д. На основе таких требований строится

множество F допустимых операторов преобразования мира, которые фактически являются совокупностью разрешенных действий робота. Желаемое расположение кубиков на площадках есть целевое состояние мира, решением задачи является последовательность действий робота (цепочка операторов $F_1, F_2, \ldots, F_i$), с помощью которой можно переставить кубики из некоторого начального расположения (начальное состояние мира) в желаемое, т. е. целевое состояние.



Три пронумерованных кубика располагаются на трех площадках. Манипуляционный робот может перемещать с одной площадки на другую по одному кубику. Кубики могут устанавливаться друг на друга. Для перемещения разрешается брать только верхний кубик. Заданы начальное и целевое расположение кубиков. Задача заключается в определении плана перемещения кубиков, позволяющего за минимальное число шагов (шаг — одно перемещение кубика) перейти от начального расположения к целевому.

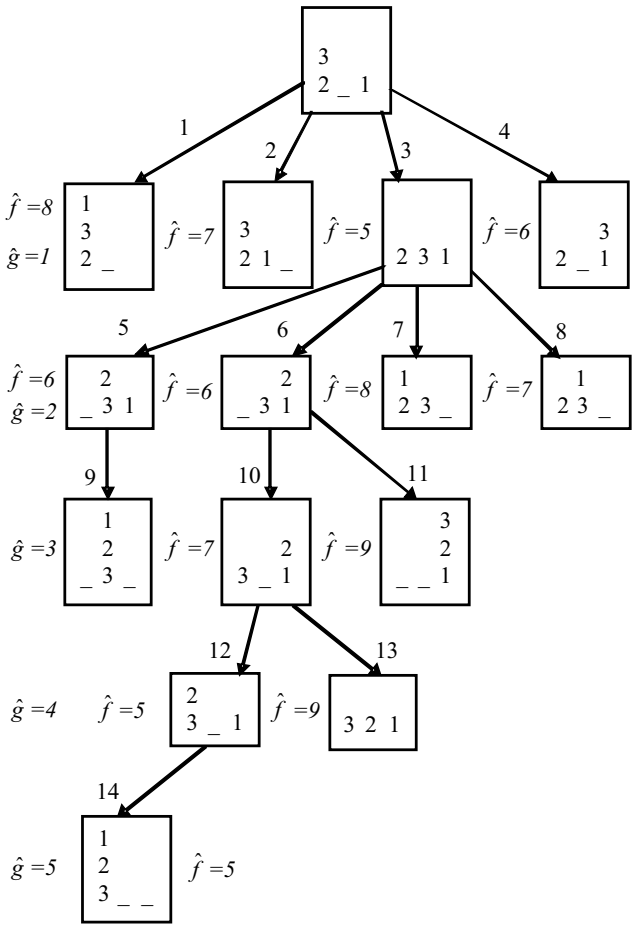


Рис. 6 – Граф решения задачи преобразования сцен.

В качестве эвристической функции $\hat{h}(v)$ примем сумму числа кубиков, расположенных не на своем месте, и общего числа кубиков, препятствующих установлению каждого из них на свое место. Препятствующими считаются кубики, расположенные сверху какого-либо кубика, который нужно переставить, и кубики, которые занимают «чужие» места в целевой конфигурации, т. е. места, в которые должны быть установлены кубики, расположенные не на своем месте.

Граф решения задачи с помощью алгоритма упорядоченного перебора при указанной эвристической функции приведен на рис. 6.

Рассмотрим некоторые свойства эвристических алгоритмов упорядоченного перебора, использующих оценочную функцию $\hat{f}(v) = \hat{g}(v) + \hat{h}(v)$. Они являются гарантирующими, если для всех вершин v выполняется условие $\hat{h}(v) \leqslant h(v)$ и стоимости всех дуг превосходят некоторое минимальное число.

Для сравнения эффективности алгоритмов перебора используется понятие эвристической силы. Пусть A_1 и A_2 — произвольные гарантирующие алгоритмы перебора, а $f_1 = g(v)+\hat{h}_1(v)$ и $f_2 = g(v)+\hat{h}_2(v)$ — используемые ими оценочные функции. Алгоритм A_1 называют эвристически более сильным, чем алгоритм A_2, если на всех вершинах, кроме целевой, выполняется неравенство $\hat{h}_1(v) > \hat{h}_2(v)$. Эвристически более сильный алгоритм A_1 раскрывает меньшее число вершин при поиске минимального пути, чем алгоритм A_2.

Обозначим B множество гарантирующих алгоритмов, которые можно использовать для решения данной задачи. Алгоритм $A^* \in B$ называется оптимальным, если он не раскрывает большее число вершин, чем любой другой алгоритм $A \in B$.

Оптимальность в указанном смысле не является исчерпывающей характеристикой эффективности алгоритма, т. к. не учитывает сложность вычисления оценки эвристической функции $\hat{h}(v)$. В большинстве практических задач более объективной характеристикой является оценка объема вычислений, необходимых для определения значений $\hat{h}(v)$ на всех шагах к целевой вершине.

Поиск решений при сведении задач к подзадачам

Поиск при редукции задач. Поиск решения при данном типе представления задачи опирается на граф редукции задачи, который является графом типа И-ИЛИ. Неявно граф И-ИЛИ определяется посредством описания операторов порождения подзадач. Оператор преобразует исходное описание задачи во множество дочерних подзадач. Это преобразование должно быть таким, чтобы решение всех дочерних подзадач обеспечивало решение исходной задачи. Если множество дочерних задач состоит из одного элемента, то производится замена одной задачи другой, эквивалентной ей. Для конкретной задачи может существовать множество операторов преобразования. Применение каждого оператора порождает альтернативное множество подзадач, что определяет существование отношения ИЛИ на графе редукции.

Построение графа редукции задачи аналогично построению графа поиска решения в пространстве состояний. Цель поиска — показать, что начальная вершина разрешима. Процедуру поиска можно интерпретировать как построение дерева решения. Дерево решения — это поддерево (подграф) графа редукции задачи с корнем в начальной вершине, состоящее из разрешимых вершин.

Поиск на графе редукции задачи отличается от поиска на графе состояний тем, что он включает процедуры проверок разрешимости и неразрешимости вершин вместо процедуры проверок соответствия состояния целевому.

Вершина неразрешима, если она является тупиковой или если неразрешимы все ее дочерние вершины, когда они связаны отношением ИЛИ, или хотя бы одна из дочерних вершин, когда они связаны отношением И. Редукция (раскрытие вершин) заканчивается, если устанавливается разрешимость или неразрешимость начальной вершины.

Как и при поиске в пространстве состояний различают методы слепого и упорядоченного перебора на графе редукции [4, 7].

Алгоритм полного перебора. В методе полного перебора вершины раскрываются в том порядке, в котором они строятся. Алгоритм полного перебора при поиске решающего графа имеет специфику, обусловленную процедурами проверок, являются ли вершины разрешимыми или неразрешимыми. Он формируется следующим образом:

1) поместить начальную вершину S_0 в список ОТК;
2) взять первую вершину из списка ОТК и поместить ее в список ЗКР; обозначить эту вершину через v;
3) раскрыть вершину v и поместить все ее дочерние вершины в конец списка ОТК и провести от них указатели к вершине v. Если дочерних вершин не оказалось, то поместить вершину как неразрешимую и перейти к следующему шагу; в ином случае перейти к шагу 7;
4) применить к дереву поиска процедуру разметки неразрешимых вершин;
5) если начальная вершина помечена как неразрешимая, то выдать результат о неудаче; в ином случае перейти к следующему шагу;
6) изъять из списка ОТК все вершины, имеющие неразрешимые предшествующие им вершины, и перейти к шагу 2;
7) если дочерние вершины являются связанными, то перейти к следующему шагу; в ином случае перейти к шагу 9;
8) если все дочерние вершины являются заключительными, то поместить v как разрешимую и перейти к 10, в ином случае — как неразрешимую и идти к шагу 4;
9) если хотя бы одна дочерняя вершина является заключительной, то поместить как разрешимую и идти к 10, иначе — как неразрешимую и перейти к 4;
10) произвести разметку разрешимых вершин;
11) если начальная вершина помечена как разрешимая, то выдать дерево решения; в ином случае перейти к следующему шагу;
12) изъять из списка ОТК все вершины, являющиеся разрешимыми или имеющие разрешимые предшествующие им вершины, и перейти к шагу 2.

Рисунок 7 — это пример поискового дерева редукции задачи, на вершинах которого приведены цифры, указывающие последовательность раскрытия вершин при использовании алгоритма полного перебора. Заключительные вершины помечены буквой Z. Дерево решения выделено жирной линией. Вершины A и B не

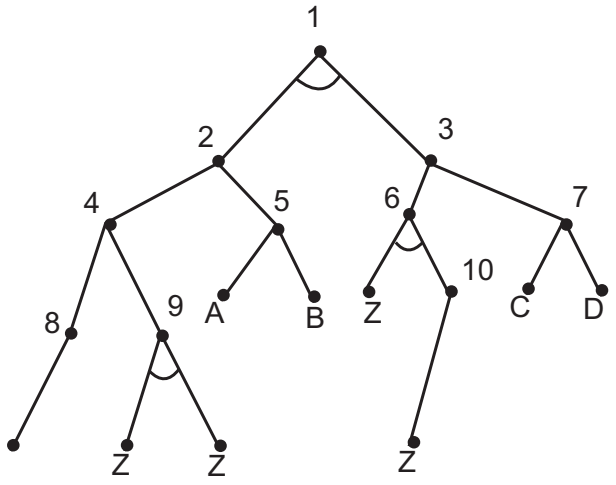


Рис. 7 – Дерево состояний алгоритма полного перебора.

раскрыты, т. к. являются тупиковым. Вершины C и D могут быть не тупиковыми и не заключительными, но они не раскрыты, т. к. по мере раскрытия вершины 10 найдено решающее дерево и алгоритм полного перебора прекращает работу.

Алгоритм перебора в глубину. При методе перебора в глубину ищется дерево решения в пределах заданной граничной глубины, и каждый раз раскрывается та из вершин с глубиной, строго меньшей граничной глубины, которая построена последней. Если решающее дерево содержит вершины с глубиной, превышающей граничную глубину, естественно, что решение не будет найдено. Алгоритм перебора в глубину включает такую последовательность шагов:

1) поместить начальную вершину S_0 в список ОТК;
2) взять первую вершину из списка ОТК и поместить ее в список ЗКР; обозначить эту вершину через v;
3) если глубина вершины v равна граничной глубине, то отметить вершину v как неразрешимую и перейти к шагу 5; в ином случае перейти к следующему шагу;
4) раскрыть вершину v. Поместить все дочерние вершины (в произвольном порядке) в начало списка ОТК и провести от них указатели к вершине v. Если дочерних вершин не оказалось, то пометить вершину v как неразрешимую и перейти к следующему шагу, в ином случае перейти к 8;
5) применить к дереву поиска процедуру разметки неразрешимых вершин;
6) если начальная вершина помечена, как неразрешимая, то выдать результат о неудаче; в ином случае перейти к следующему шагу;
7) изъять из списка ОТК все вершины, имеющие неразрешимые предшествующие им вершины. Перейти к шагу 2;
8) если все дочерние вершины являются связанными, то перейти к следующему шагу; в ином случае перейти к шагу 10;
9) если все дочерние вершины являются заключительными, то пометить v как разрешимую и перейти к 11, иначе — как неразрешимую и идти к 5;

10) если хотя бы одна дочерняя вершина является заключительной, то пометить v как разрешимую и идти к 11, иначе — как неразрешимую и перейти к 5;
11) применить к дереву перебора процедуру разметки разрешимых вершин;
12) если начальная вершина помечена как разрешимая, то выдается дерево решения; в ином случае перейти к следующему шагу;
13) изъять из списка ОТК все вершины, являющиеся разрешимыми или имеющие разрешимые предшествующие им вершины, и перейти к шагу 2.

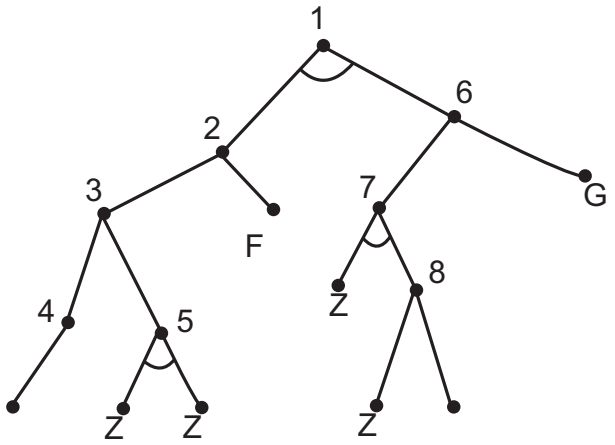


Рис. 8 – Дерево состояний алгоритма перебора в глубину.

На рис. 8 приведено поисковое дерево редукции задачи, которое строится при переборе в глубину с граничной глубиной, равной 4. Жирной линией выде-лено решающее дерево. Цифры на вершинах указывают на последовательность, в которой раскрываются вершины. Вершина F не раскрыта, хотя она не является ни тупиковой, ни заключительной и не достигла граничной глубины, т. к. до наступления момента ее раскрытия становится известным, что предшествующая ей вершина 2 разрешима и в соответствии с шагом 11 она изымается из списка ОТК. Вершина G не раскрыта, т. к. решающее дерево найдено.

При переходе на дерево И-ИЛИ возникает особенность при появлении повторяющихся вершин, т. е. вершин, эквивалентных (определяющих одну и ту же подзадачу) ранее построенным. В модифицированных методах полного перебора и перебора в глубину на дереве состояний повторяющиеся вершины вновь не строятся. При переборе на дереве типа И-ИЛИ этого делать нельзя (рис. 9).

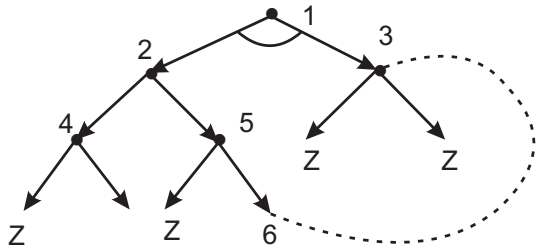


Рис. 9 – Пример поискового дерева типа И-ИЛИ.

Пример поискового дерева типа И-ИЛИ, построенного в процессе полного перебора. На этом дереве вершина 6 является повторяющейся — она эквивалентна вершине 3. Очевидно, повторяющуюся вершину 6 не строить нельзя, т. к. в ином случае не удалось бы получить решающее дерево. Если в процедуре предусмотрена процедура установления эквивалентности различных вершин, то в данном примере повторяющуюся вершину 6 раскрывать не обязательно, т. к. к моменту ее раскрытия известно, что эквивалентная ей вершина 3 разрешима.

Эвристические методы поиска (перебора). Для упорядочивания раскрываемых вершин на дереве типа И-ИЛИ используется так называемое оптимальное потенциальное дерево решения, для выделения которого применяется эвристическая функция. При поиске решения в пространстве состояний эвристическая функция определялась как оценка стоимости оптимального пути от заданной вершины до целевой. В деревьях типа И-ИЛИ для ее определения используется оценка стоимости дерева решения.

Стоимость дерева решения. Различают суммарную стоимость, представляющую собой сумму стоимости всех дуг в дереве решения и максимальную стоимость, равную стоимости пути между двумя вершинами дерева решения, имеющего максимальную стоимость. Стоимость пути определяется как сумма стоимостей дуг, входящих в этот путь. Эти определения можно пояснить на примере дерева решения (жирная линия), где цифры около дуг указывают на стоимости. В этом примере суммарная стоимость равна 20, максимальная стоимость — 15.

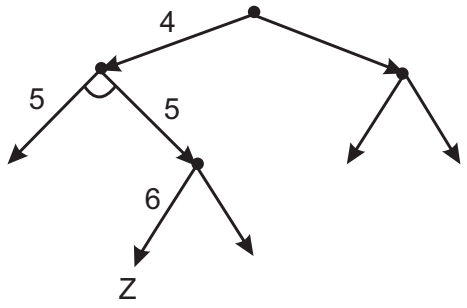


Рис. 10 – Дерево решения редукционного графа.

Если стоимость дуг равна единице, то суммарная стоимость равна числу дуг в дереве решения, а максимальная стоимость — числу дуг в пути между двумя самыми отдаленными вершинами дерева решения.

Дерево решения, имеющее минимальную стоимость (суммарную или максимальную в зависимости от того, какая из них принята за критерий оптимальности), называется оптимальным.

Эвристическая функция. Так называется функция h, являющаяся оценкой минимальной стоимости $\hat{h}(v)$. Рассмотрим, как может быть построена эвристическая функция. Дерево поиска, которое строится в ходе процесса перебора на каждом этапе задачи, пока не закончено построение дерева редукции, обладает вершинами, не имеющими построенных дочерних вершин. Такие вершины называются *концевыми*. Они могут быть заключительными, тупиковыми и нераскрытыми, т. е. вершинами, для которых на рассматриваемом этапе еще не построены дочерние вершины. Если концевая вершина v — заключительная вершина, то $h(v) = 0$; если

v — нераскрытая вершина, то $h(v)$ строится как оценка функции $\hat{h}(v)$ на основе эвристической информации, связанной с v исходной задачей.

Для не концевых вершин эвристическая функция строится так:

- если v имеет несвязанные дочерние вершины $v_1, \ldots, v_k$, то

$$\hat{h}(v) = \min_{i \in (l,\ldots,k)} \left[c(v, v_i) + \hat{h}(v_i) \right],$$

- если v имеет связанные дочерние вершины $v_1, \ldots, v_k$, то эвристические функции для суммарной и максимальной стоимостей определяются соответственно формулами:

$$\hat{h}(v) = \sum_{i=1}^{k} \left[c(v, v_i) + \hat{h}(v_i) \right],$$
$$\hat{h}(v) = \max_{i \in (l,\ldots,k)} \left[c(v, v_i) + \hat{h}(v_i) \right].$$

Для тупиковых вершин функция $\hat{h}(v)$ не определена.

Эвристический алгоритм упорядоченного перебора. В дереве поиска содержится множество поддеревьев с корнем в начальной точке, каждое из которых могло бы быть начальной частью дерева решения. Эти поддеревья называют потенциальными деревьями решения. Потенциальное дерево решений D_0 назовем оптимальным в том случае, когда оно обладает следующим свойством: если у вершины v, входящей в дерево D_0, в дереве перебора имеются связанные дочерние вершины, то все они входят в дерево D_0; если у этой вершины в дереве перебора имеются несвязанные дочерние вершины $v_i, (i = l, \ldots, k)$, то в дерево D_0 входит та из них, для которой значение $\left[c(v, v_1) + \hat{h}(v_1) \right]$ минимально.

При эвристическом методе упорядоченного перебора предполагается, что оптимальное потенциальное дерево решений является начальной частью оптимального дерева решения и его концевые нераскрытые вершины раскрываются в первую очередь. Для того чтобы в ходе перебора можно было извлечь из дерева поиска дерево D_0, необходимо после очередного раскрытия вершины вычислять эвристическую функцию $\hat{h}$ для каждой из вновь построенных вершин, предшествующих только что раскрытой вершине. Для остальных вершин значение $\hat{h}$ не меняется.

Эвристический алгоритм упорядоченного перебора можно сформулировать следующим образом.

1) Поместить начальную вершину в список ОТК и вычислить $\hat{h}(S_0)$.
2) Выделить оптимальное потенциальное дерево решений D_0.
3) Выбрать некоторую концевую вершину дерева D_0, которое имеется в списке ОТК, и поместить в список ЗКР. Обозначить эту вершину через v.
4) Если v — заключительная вершина, то пометить ее как разрешимую и перейти к следующему шагу. В ином случае перейти к шагу 8.
5) Применить к дереву D_0 процедуру разметки разрешимых вершин.
6) Если начальная вершина помечена как разрешимая, то выдать дерево D_0 в качестве дерева решения; иначе перейти к следующему шагу.

7) Изъять из списка ОТК все вершины, имеющие разрешимые предшествующие им вершины, и перейти к шагу 2.
8) Раскрыть вершину v. Если дочерних вершин она не имеет, то пометить ее как неразрешимую и перейти к следующему шагу; в ином случае перейти к шагу 12.
9) Применить к дереву D_0 процедуру разметки неразрешимых вершин.
10) Если начальная вершина помечена как неразрешимая, то выдать результат о неудаче; в ином случае перейти к следующему шагу.
11) Изъять из списка ОТК все вершины, имеющие неразрешимые предшествующие им вершины, и перейти к шагу 2.
12) Поместить все дочерние вершины в список ОТК и провести от них указатели назад к вершине S. Вычислить $\hat{h}$ для этих вершин. Пересчитать величины $\hat{h}$ для вершины V и для предшествующих ей вершин.
13) Перейти к шагу 2.

3.2 Поиск в иерархических и альтернативных пространствах

Методы поиска в одном пространстве не позволяют решать сложные задачи, так как с увеличением размера пространства время поиска экспоненциально растет. При большом размере пространства поиска можно попробовать разбить общее пространство на подпространства и осуществить поиск сначала в них. Пространство поиска представлено иерархией пространств.

Методы поиска решения в иерархических пространствах обычно делятся:

1) на поиск в факторизованном пространстве;
2) поиск в фиксированном множестве пространств;
3) поиск в изменяющемся пространстве множеств.

Поиск в факторизованном пространстве. Во многих приложениях требуется найти все решения. Например — постановка диагноза. Пространство называется факторизованным, если оно разбивается на непересекающиеся подпространства (классы) частичными (неполными) решениями. Причем по виду частичного решения можно определить, что оно не приведет к успеху, т. е. что все полные решения, образованные из него, не приведут к целевым решениям. Поиск в факторизованном пространстве осуществляется на основе метода «иерархическая генерация-проверка». Если пространство поиска удается факторизовать, то поиск даже в очень большом пространстве можно организовать эффективно.

Поиск в фиксированном множестве пространств. Применение метода факторизации пространства ограничено тем, что для ряда областей не удается по частичному решению сделать заключение о его непригодности. Например задачи планирования и конструирования. В этих случаях могут быть применены методы поиска, использующие идею абстрактного пространства. Абстракция должна подчеркнуть важные особенности рассматриваемой задачи, позволить разбить задачу на более простые подзадачи и определить последовательность подзадач (план решения), приводящую к решению основной задачи.

Поиск в изменяющемся множестве иерархических пространств. В ряде приложений не удается все решаемые задачи свести к фиксированному набору подзадач.

План решения задачи в данном случае должен иметь переменную структуру и не может быть сведен к фиксированному набору подзадач. Для решения подобных задач может быть использован метод нисходящего уточнения. Этот метод базируется на следующих предположениях:

1) возможно осуществить частичное упорядочение понятий области, приемлемое для всех решаемых задач;
2) решения, принимаемые на верхних уровнях, нет необходимости отменять на более нижних.

Поиск в альтернативных пространствах. Рассмотренные выше методы поиска исходят из молчаливой предпосылки, что знания о предметной области и данные о решаемой задаче являются точными и полными и для них справедливо следующее:

1) все утверждения, описывающие состояние, являются истинными;
2) применение оператора к некоторому состоянию формирует некоторое новое состояние, описание которого состоит только из истинных фактов.

Однако при решении любых практических задач и особенно при решении неформализованных задач распространена обратная ситуация. Эксперту приходится работать в условиях неполноты и неточности знаний (данных) и, как правило, в условиях дефицита времени. Когда эксперт решает задачу, он использует методы, отличающиеся от формальных математических рассуждений. В этом случае эксперт делает правдоподобные предположения, которые он не может доказать; тем самым вопрос об их истинности остается открытым. Все утверждения, полученные на основе этих правдоподобных предположений, также не могут быть доказаны.

Выводы

Итак, для того чтобы система могла делать умозаключения, основанные на здравом смысле, при работе с неполными (неточными) данными и знаниями, она должна быть способна делать предположения, а при получении новой информации, показывающей ошибочность предположений, отказываться как от сделанных предположений, так и от умозаключений, полученных на основе этих предположений.

Мнение системы о том, какие факты имеют место, изменяется в ходе рассуждения, т. е. можно говорить о ревизии мнений. Таким образом, даже если рассматривать проблемную область как статическую, неполнота (и неточность) знаний и данных влечет за собой рассмотрение этой области при различных (и даже противоположных) предположениях, что, в свою очередь, приводит к представлению области в виде альтернативных пространств, соответствующих различным, возможно, противоречивым и (или) взаимодополняющим предположениям и мнениям.

Все неудачи, возникшие при поиске в одном направлении, не запоминаются при переходе к поиску в другом направлении. Та же самая причина неудачи может заново обнаруживаться и на новом направлении.

Осуществлять возврат целесообразно не к состоянию, непосредственно предшествующему данному, а к тому состоянию, которое является причиной возникновения неудачи. В используемых нами терминах причиной неудач являются предположения, т. е. недоказуемые утверждения. Поэтому при обнаружении неудачи необходимо возвращаться в состояние, где это предположение было сделано, и испытывать другое предположение.

Этот метод поиска называют поиском, направляемым зависимостью.

Поиск с использованием нескольких моделей. Все методы поиска, рассмотренные до сих пор, использовали при представлении проблемной области какую-то одну модель, т. е. рассматривали область с какой-то одной точки зрения. При решении сложных задач в условиях ограниченных ресурсов использование нескольких моделей может значительно повысить мощность системы. Объединение в одной системе нескольких моделей дает возможность преодолеть следующие трудности.

1) Переход с одной модели на другую позволяет обходить тупики, возникающие при поиске в процессе распространения ограничений.
2) Использование нескольких моделей позволяет в ряде случаев уменьшить вероятность потери хорошего решения (следствие неполного поиска, вызванного ограниченностью ресурсов) за счет конструирования полного решения из ограниченного числа частичных кандидатов путем их расширения и комбинации.
3) Наличие нескольких моделей позволяет системе справляться с неточностью (ошибочностью) данных.

Контрольные вопросы

1) Опишите классификацию задач искусственного интеллекта по общим признакам.
2) Представьте классификацию задач искусственного интеллекта по типу решаемой задачи.
3) Перечислите способы представления задач.
4) Поясните, какие два этапа необходимы для процесса решения задач.
5) Расскажите о формах описания состояний и операторов при поиске решения задачи.
6) Какие два типа структур взаимосвязи задач вы знаете?
7) Приведите примеры деревьев и преобразованных деревьев редукции задачи.
8) Какие особенности предметной области и требования позволяют правильно выбрать метод решения задачи?
9) Перечислите методы решения задач в одном пространстве.
10) Опишите поиск решения в пространстве состояний и приведите пример.

11) Опишите алгоритмы полного перебора (в ширину), в глубину, упорядоченного перебора решения задач в пространстве состояний.
12) Приведите примеры решения задач по разным алгоритмам в пространстве состояний.
13) Опишите поиск решения задач при редукции задачи на подзадачи.
14) Чем отличается поиск решений на графе редукции задачи от поиска на графе состояний?
15) Сформулируйте алгоритмы поиска решений при редукции задачи.
16) Представьте поисковое дерево редукции задачи для выбранного вами примера.
17) Как определяется эвристическая функция при поиске решения в пространстве состояний?
18) Как определяется эвристическая функция при поиске решения при редукции задачи на подзадачи?
19) Определите суммарную и максимальную стоимость, т. е. стоимость дерева решения в И-ИЛИ графе.
20) Рассмотрите пример дерева решения редукционного графа и определите оптимальное дерево решения для начальной вершины.
21) Перечислите методы поиска решения задачи в иерархических пространствах.
22) Дайте характеристику факторизованного пространства.
23) Опишите поиск решения задачи в фиксированном множестве пространств.
24) На чем базируется поиск решений задачи в изменяющемся множестве иерархических пространств.
25) Опишите поиск решения задачи в альтернативных пространствах.
26) Проанализируйте преимущества поиска решения задачи с использованием нескольких моделей.

Литература

[1] Алиев Р. А. Производственные системы с искусственным интеллектом / Р. А. Алиев, Н. М. Абдикеев, М. М. Шахназаров. — М. : Радио и связь, 1990. — 246 с.

[2] Вагин В. Н. Дедуктивный вывод на знаниях // Искусственный интеллект : справочник / под ред. Д. А. Поспелова. — М. : Радио и связь, 1990. — Кн. 2. : Модели и методы — С. 89–105.

[3] Назаров В. М. Техническая имитация интеллекта: учеб. пособие для вузов / В. М. Назаров, Д. П. Ким, И. М. Макрова. — М. : Высш. шк., 1998. — 144 с.

[4] Нильсон Н. Искусственный интеллект. Методы поиска решений: пер. с англ. / Н. Нильсон. — М. : Мир, 1973. — 270 с.

[5] Нильсон Н. Проблемы искусственного интеллекта: пер. с англ. / Н. Нильсон. — М. : Радио и связь, 1985. — 280 с.

[6] Нильсон Н. Принципы искусственного интеллекта: пер. с англ. / Н. Нильсон. — М. : Радио и связь, 1985. — 375 с.

[7] Попов Э. В. Алгоритмические основы интеллектуальных роботов и искусственный интеллект / Э. В. Попов, Г. Р. Фирдман. — М. : Наука, 1976. — 456 с.

[8] Тельнов Ю. Ф. Интеллектуальные информационные системы в экономике / Ю. Ф. Тельнов. — М. : Московский государственный университет экономики, статистики и информатики, 1998. — 174 с.

[9] Newell A., Siwon H. GPS: A Program that Simulates Human Thought / Ed. by Feigenbaum E. A. and Feldman J. // Computers and Thought. — №4: McGraw-Hill, 1963.

[10] Allen J. AI Growing up / J. Allen // AI MAGAZINE. — 1998. — V. 19. — №4. — P. 13–23.

[11] Russell S. L. Artificial intelligence: a modern approach / S. L. Russell, P. Norvig. — Upper Saddle River, New Jersey: Prentice — Hall Inc., 1995. — 905 p.