11. РЕШЕНИЕ ЗАДАЧ: ПРОЦЕДУРЫ ПОИСКА РЕШЕНИЙ

Решение задач является фундаментальным для большинства приложений ИИ. Фактически, способность решать задачи часто используется как мера интеллектуальности как для людей, так и для машин. Существует два типа задач. Задачи первого типа могут быть решены, с использованием детерминированной процедуры, которая будет гарантированно успешной. Для задач данного типа существуют алгоритмы решения, как, например, в математике**:** решение уравнений, методы оптимизации и т.п. Эти алгоритмы (процедуры решения) преобразуются в программы, которые выполняет компьютер. Однако мало реальных задач могут быть решены таким способом. Более обширный класс задач составляет вторую категорию, то есть задачи, которые могут быть решены методом поиска решения. Этот подход к решению задач являются предметом ИИ.

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

Задача считается хорошо определенной, *если для неё имеется возможность задать пространство возможных решений (состояний), а также способ просмотра этого пространства с целью поиска конечного (целевого) состояния, соответствующего решенной задаче*. Поиск конечного состояния задачи заключается в применении алгоритмической процедуры перехода от одного состояния к другому и проверки, не является ли текущее состояние решением задачи. Поисковая процедура продолжается до тех пор, пока не будет найдено решение. Примерами хорошо определенных задач являются: доказательство теорем, поиск маршрута на карте, планирование действий робота в среде с препятствиями, различные игры с полной информацией и др.

Человек обычно не решает задачу в той форме, в которой она изначально формулируется. Он стремится представить задачу таким образом, чтобы ему было удобно 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 важность. 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; способ представления должен быть таким, чтобы решателю задач было удобно с ним работать.

Для решения задач на ЭВМ наиболее часто используются следующие способы представления:

- 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. Исследования 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. Эти 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.

Одно из наиболее трудных препятствий, которое необходимо преодолевать при попытках применения методов ИИ к реальным задачам - это огромная размерность и сложность большинства ситуаций. На начальном этапе развития исследований в области ИИ первостепенное значение имела разработка хороших методов поиска для того, чтобы можно было решать эти задачи на сильно ограниченных по возможностям компьютерах, которые использовались в то время. К тому же, нужны были хорошие методы поиска, потому что всегда полагали, что поиск является основным элементом решения задачи и он - наиболее важный компонент интеллекта.

11.1. Введение в проблему и терминология

Представим, что некто потерял ключи от автомобиля. Он знает, что ключи где-то в доме, который выглядит как показано на рисунке 11.1.

Буква «х» показывает, что отсутствие ключей обнаружилось у входной двери. Поиск начинается с проверки гостиной комнаты. Затем через холл бедолага проходит в комнату 1, после чего возвращается в холл и идет во вторую комнату. После этого он возвращается в холл и идет в комнату 3 (master bedroom). Затем возвращается и, проходя гостиную комнату, идет в кухню, где и находит потерянные ключи. На рисунке 11.2. изображен граф пути поиска ключей.

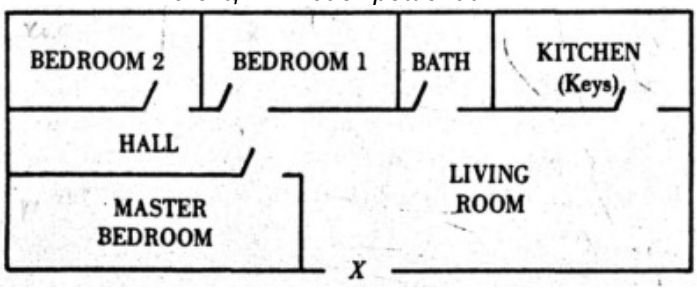

Рисунок 11.1 План квартиры

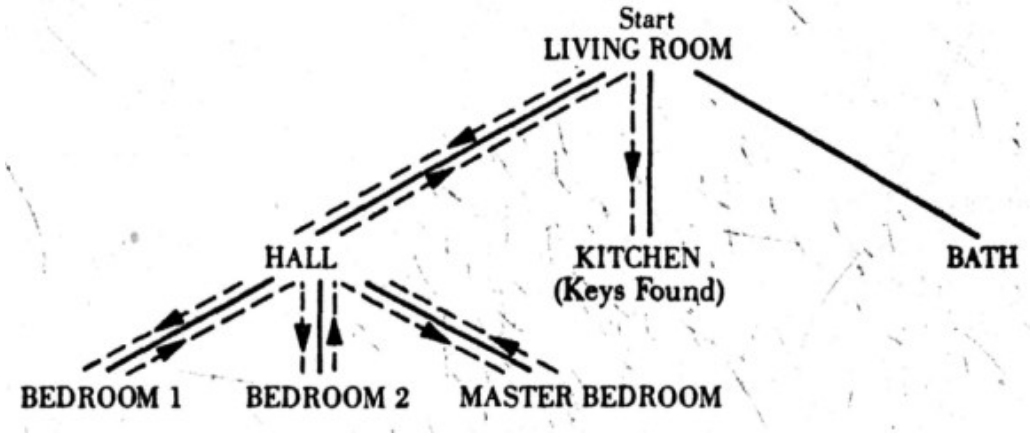

Рисунок 11.2 Граф возможных путей нахождения потерянных ключей

Изображение решения задач в виде графа, дает простое наглядное представление о том, как работают различные методы поиска, а также позволяет исследователю применять различные положения теории графов.

Основные определения:

Узел: Отдельный пункт и возможная цель.

Терминальный узел: Узел, который заканчивает путь.

Область поиска: Множество всех узлов.

Цель: Узел, который является объектом поиска.

Эвристика: Информация о вероятности того, что некоторый определенный узел является более предпочтительным выбором, чем другой.

Путь решения: Направленный граф пройденных узлов и дуг, которые ведут к решению.

В примере с потерянными ключами каждая комната в доме - узел; весь дом - область поиска; цель - кухня; путь решения - граф, который показан на рис. 11.2. Комнаты и ванна - терминальные узлы, потому что они не ведут куда-либо дальше. В этом примере эвристика не использовалась.

11.2. Комбинаторный взрыв

Для большинства реальных задач область поиска состоит из большого числа узлов. Вместе с увеличением области поиска увеличивается и число различных возможных путей к цели. Проблема состоит в том, что каждый узел, который добавляется к области поиска, увеличивает число путей более чем на один. Поэтому увеличение количества путей происходит быстрее, чем увеличение количества узлов. Простой пример из комбинаторики. Для трех букв возможны шесть различных перестановок:

A B C
A C B
B C A
B A C
C B A
C A B

Это все способы, которыми можно переставить буквы A, B и C. Количество перестановок можно получить по формуле из комбинаторики**:** число способов, которыми N объектов могут быть переставлены, равняется N! (N факториал). Факториал числа это результат перемножения всех целых чисел от N до 1. Поэтому 3! равняется 3 * 2 * 1 = 6. Теперь можно увидеть, что добавление 1 объекта увеличит количество перестановок до 24 (4! = 24). С 5 объектами число их равно 120; с 6 объектами - 720 и т.д. Для 1000 объектов число

возможных перестановок просто огромно. *Ситуацию, когда малое увеличение количества объектов вызывает резкое увеличение количества комбинаций*, называют **комбинаторным взрывом**. При достаточно большом количестве объектов обследовать все комбинации становится практически невозможным, их будет трудно даже перечислить.

Комбинаторный взрыв обнаруживается и при решении задач. Поскольку каждый узел, добавляемый в область поиска, увеличивает число возможных решений более чем на 1 то, наступает момент, когда путей, которые нужно анализировать, становится слишком много. Поэтому только для самых простых задач возможно провести исследование всех путей (исчерпывающий поиск или полный перебор, когда исследуются все узлы).

Теоретически метод полного перебора, или "грубой силы" работает всегда, но на практике это требует слишком больших временных или вычислительных ресурсов, либо и того и другого вместе. По этой причине были разработаны другие технологии поиска.

11.3. Общая характеристика представления задач в пространстве состояний

Представление задач в пространстве состояний предполагает задание ряда описаний: состояний, множества операторов и их воздействий на переходы между состояниями, целевых состояний. Описания состояний могут представлять собой строки символов, векторы, двухмерные массивы, деревья, списки и т. п. Операторы переводят одно состояние в другое. Иногда они представляются в виде продукций A=>B, означающих, что состояние *A* преобразуется в состояние *B*.

Пространство состояний можно представить как граф, вершины которого помечены состояниями, а дуги - операторами.

Подход, использующий пространство состояний, включает в себя:
- задание начальных состояний задачи;
- задание конечных (целевых) состояний задачи;
- задание операторов, преобразующих одни состояния задачи в другие.

Решение задачи состоит в том, чтобы найти последовательность операторов, преобразующих начальное состояние задачи в конечное, которое соответствует решенной задаче. Таким образом, проблема поиска решения задачи <*A, B*> представляется, как проблема поиска на графе пути из *A* в *B*. Обычно графы не задаются, а генерируются по мере надобности.

Существуют многочисленные задачи, решение которых интерпретируется как поиск в пространстве состояний. Вот примеры некоторых задач.

Поиск пути в сети улиц. Задача состоит в том, чтобы на схеме улиц найти путь из начального пункта в конечный пункт. Задание только топологии сети улиц не полностью описывает задачу. Необходимо указать, в чем должно состоять решение задачи, и какие ограничения накладываются на допустимые переходы между состояниями задачи.

Для данной задачи состояние определяется положением перемещающегося объекта на схеме улиц. Непосредственный переход между двумя состояниями допустим, если между соответствующими позициями на схеме имеется прямая связь (т.е. улица). Начальное и конечное (целевое) состояния представляются определенными позициями объекта. Задача считается решенной, если проложен путь из начального пункта в конечный пункт за конечное число элементарных переходов.

Менее тривиальным является поиск пути минимальной стоимости между начальным и целевым пунктом. Затратами, учитываемыми при поиске такого пути, могут быть: расстояние, время, потребление бензина и т.п.

Игра-головоломка "8". Игра-головоломка "8" - это одна из показательных задач поиска решений в пространстве состояний. В игре используется восемь фишек, пронумерованных от 1 до 8 (рисунок 11.3). Фишки располагаются в девяти ячейках, образующих матрицу размером 3x3. Одна из ячеек всегда пустая. Любая фишка, смежная с пустой ячейкой, может быть передвинута в позицию, соответствующую пустой ячейке.

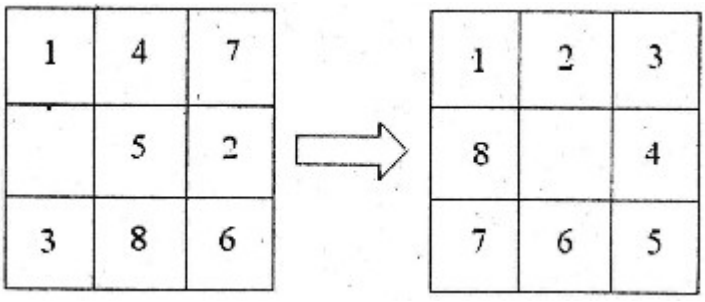

Рисунок 11.3. Игра-головоломка "8"

В головоломке ищут минимальную последовательность ходов, переводящих игру из начального состояния в конечное состояние, определяемое заранее заданной конфигурацией фишек. Для целенаправленного движения к целевому состоянию может учитываться количество неправильно расположенных фишек для каждой возможной позиции свободной ячейки.

Задача о N ферзях. На шахматной доске необходимо расставить N ферзей таким образом, чтобы они не угрожали друг другу. Расстановка ферзей начинается с пустой доски (начальное состояние) и выполняется последовательно линия за линией. Конечное состояние достигается, когда расставлены все ферзи. Смежные состояния задачи определяются условием отсутствия угрозы определенному ферзю. Для целенаправленного движения к целевому состоянию необходимо ставить очередного ферзя на соответствующую линию так, чтобы оставалось как можно больше не атакованных полей.

В общем случае задача, поставленная как задача поиска в пространстве состояний, определяется совокупностью четырех составляющих:

(S_0, S, F, G),

где

S - множество состояний;

S_0 - множество начальных состояний, $S_0 \subset \mathrm{S}$; (см. задачу о N ферзях)

F - множество операторов, преобразующих одни состояния в другие;

G - множество целевых состояний, $\mathrm{G} \subset \mathrm{S}$. (см. задачу о N ферзях)

Каждый оператор $f \in F$ является функцией, отображающей одно состояние в другое - $s_j=f(s_i)$, где s_i, $s_j \in S$. Решением задачи является последовательность операторов $f_i \in \mathrm{F}$, преобразующих начальные состояния в конечные, т.е. $f_n(f_{n-1}(\ldots(f_2(f_1(S_0)))\ldots)) \in G$. Если такая последовательность не одна и задан критерий оптимальности, то возможен поиск оптимальной последовательности.

Как было сказано, поиск решений в пространстве состояний удобно представлять в виде *графа состояний*. Множество вершин графа соответствует состояниям задачи, а множество дуг (ребер) - операторам. В этом случае поиск решения задачи может интерпретироваться как поиск пути на графе.

Уточним сказанное на примере поиска пути в лабиринте (рисунок 11.4а). Если обозначить меткой X путешественника, то начальное и конечное состояния задачи можно задать с помощью схем, изображенных на рисунке 11.4б.

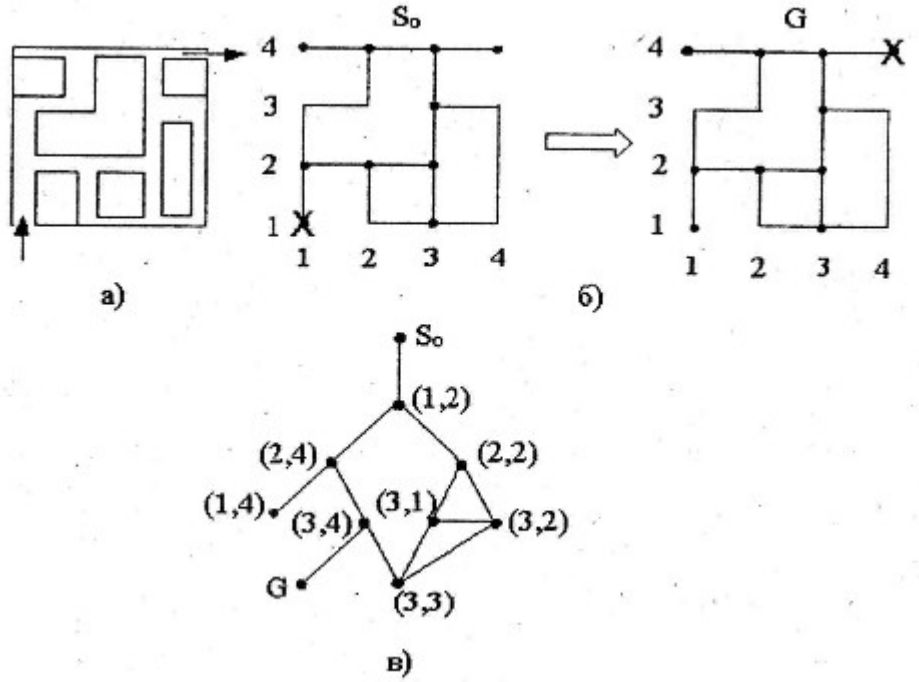

Рисунок 11.4. Поиск пути в лабиринте

Хотя путешественник может находиться в любой точке лабиринта, интерес представляют только пересечения различных путей, обозначенные точками. Попадая на пересечения путей, путешественник должен принять решение о дальнейшем направлении движения. Поэтому возможные состояния задачи определяются нахождением путешественника в указанных точках. Перемещения путешественника из одной точки в другую соответствует переходам задачи из одного состояния в другое. Следовательно, множество операторов задачи представляется только одним оператором перемещения, а решение задачи сводится к нахождению последовательности операторов перемещений, преобразующих задачу из начального состояния S_0 в конечное состояние G.

Пространство состояний рассматриваемой задачи можно представить в виде графа, изображенного на рисунке 11.4в. Вершины графа, обозначенные координатами точек пересечения, соответствуют состояниям задачи, а дуги - операторам перемещений. Так как при решении данной задачи используется только один оператор, то дуги не имеют меток. Решение задачи состоит в нахождении пути из начальной вершины S_0 в целевую вершину G.

В рассматриваемой задаче к целевой вершине ведут пять различных путей. Если количество элементарных отрезков между точками пересечений использовать в качестве оценки стоимости пути, то суммарные стоимости каждого из пяти путей будут равны: 6, 6, 8, 10, 10. В качестве оптимального пути можно выбрать один из двух путей с минимальной стоимостью.

Иногда рассматривают поиск с ограничениями. *Ограничения* представляют некоторые условия, которые должны выполняться в ходе достижения целевого состояния. Например, при поиске пути в лабиринте может быть поставлено условие, чтобы длина пройденного пути не превышала некоторого значения *L,* тогда состояние задачи будет определяться позицией путешественника в лабиринте и пройденным расстоянием.

11.4. Методы поиска

Существует несколько методов поиска решения. Наиболее важные и общие из них следующие:

* поиск в глубину
* поиск в ширину
* поиск с подъемом на холм
* поиск по наименьшей стоимости

При оценивании методов поиска следует обратить внимание на два наиболее важных вопроса:

1. Как быстро метод приводит к решению?
2. Насколько хорошим является полученное решение?

Существует ряд типов задач, в которых основная цель состоит в нахождении любого допустимого решения с минимальными усилиями. Для этих задач более важен первый критерий (вопрос). В других ситуациях, важным является то, что решение должно быть наиболее близким к оптимальному.

Длина пути к решению и фактическое число пройденных узлов определяет скорость поиска. Возврат из тупиков – по существу потраченное впустую усилие, поэтому желательно, чтобы возвратов было как можно меньше.

Важно понять различие между нахождением оптимального решения и нахождением "хорошего" решения. Поиск оптимального решения часто влечет за собой исчерпывающий поиск (полный перебор), потому что часто это может быть единственным способом определения того, было ли найдено оптимальное решение. Однако, получение хорошего решения означает нахождение такого, которое только удовлетворяет заданному набору ограничений - независимо от того, существует или нет более лучшее решение, то есть поиск любого решения из допустимой области или выбор решения из нескольких допустимых решений.

Каждый из методов поиска будет работать в каких-то определенных ситуациях лучше, чем другие. К тому же, трудно сказать, будет ли один метод поиска всегда превосходить другой. Однако, некоторые методы поиска будут иметь большую вероятность оказаться более эффективными для некоторого среднего случая. Также, само описание задачи будет иногда помогать Вам выбрать соответствующий метод поиска.

Постановка задачи-примера

Представьте, что клиент хочет заказать билет на рейс из Нью-Йорка до Лос-Анджелеса на самолет авиакомпании "XYZ Авиалинии", причем у авиакомпании "XYZ" нет прямого рейса из Нью-Йорка до Лос-Анджелеса. Клиент настаивает на полете самолетом только компании "XYZ". Список рейсов авиакомпании "XYZ":

Из Нью-Йорка в Чикаго 1000 миль
Из Чикаго в Денвер 1000 миль
Из Нью-Йорка в Торонто 800 миль
Из Нью-Йорка в Денвер 1900 миль
Из Торонто в Калгари 1500 миль
Из Торонто в Лос-Анджелес 1800 миль
Из Торонто в Чикаго 500 миль
Из Денвера в Урбану 1000 миль
Из Денвера в Хьюстон 1500 миль
Из Хьюстона в Лос-Анджелес 1500 миль
Из Денвера в Лос-Анджелес 1000 миль

Можно сразу увидеть, что имеется путь из Нью-Йорка до Лос-Анджелеса на самолете "XYZ", с использованием связанных рейсов. В результате клиенту можно предоставить билет.

Задача состоит в том, чтобы написать программы, которые выполняют ту же самую процедуру, то есть поиск маршрута из Нью-Йорка до Лос-Анджелеса.

Графическое представление

Список рейсов компании "XYZ" может быть преобразован в направленный граф, показанный на рис. 11.5. Стрелки указывают направление движения.

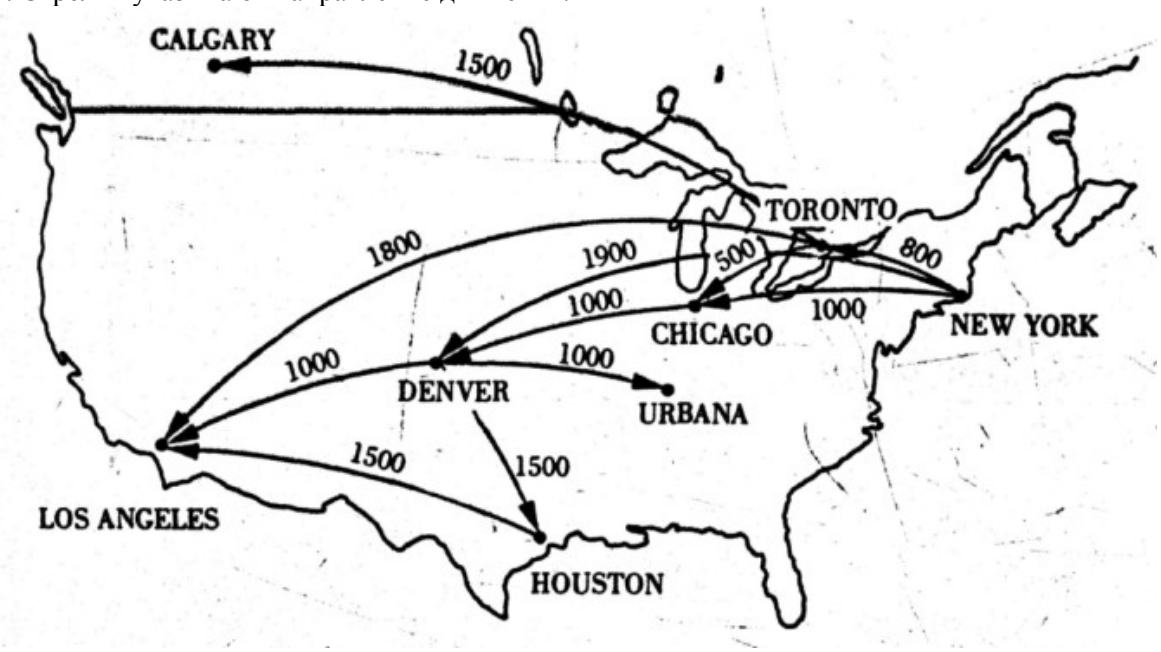

Рисунок 11.5. Направленный граф рейсов авиакомпании "XYZ Авиалинии"

Информация о рейсах будет легче восприниматься, если преобразовать граф в дерево, как показано на рис. 11.6. Здесь цель, Лос-Анджелес, обведена кружком; различные города появляются в дереве более чем один раз, чтобы упростить его построение.

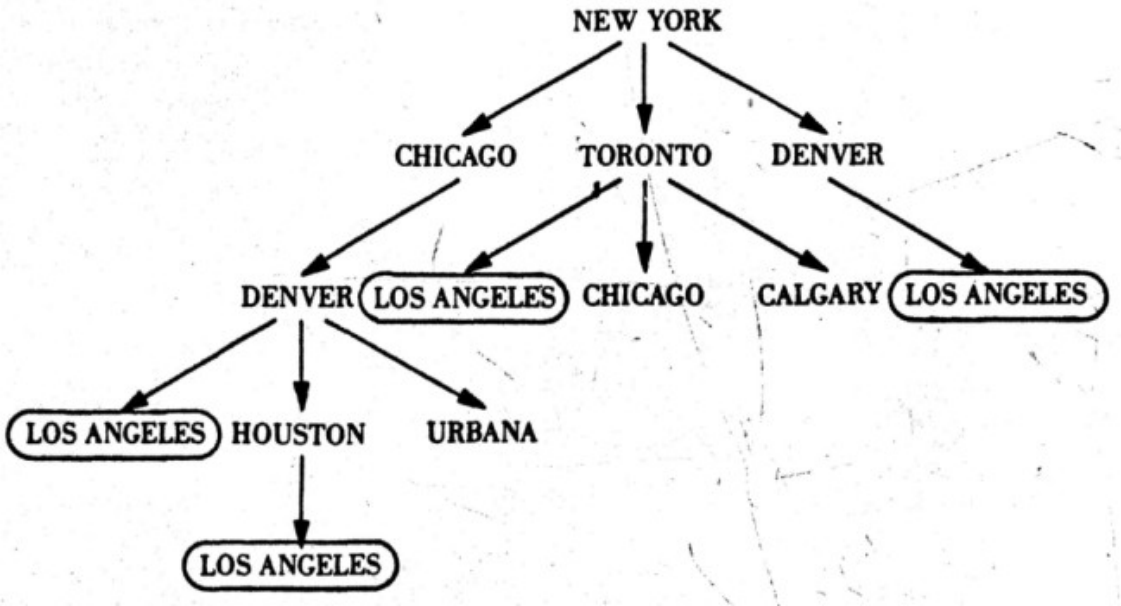

Рисунок 11.6. Версия таблицы рейсов в виде дерева.

11.4.1. «Слепые» методы поиска

Поиск в глубину

Поиск в глубину исследует каждый возможный путь до самого конца прежде, чем перейдет к другому пути, если цель не была обнаружена. Чтобы понять, как работает этот метод, рассмотрим дерево, в котором F является целью:

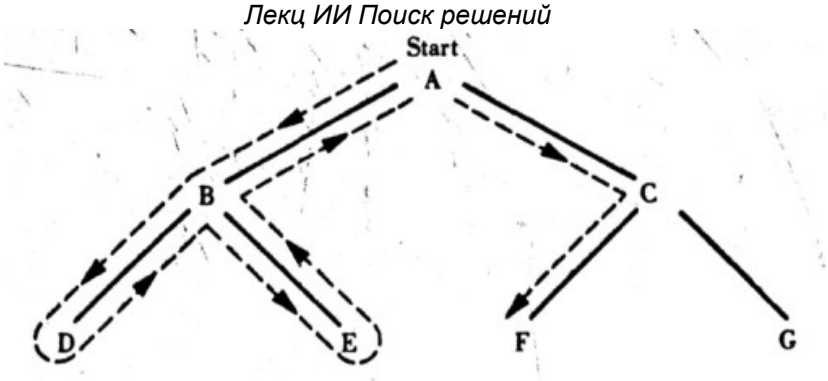

Рисунок 11.7. Путь решения (поиска цели)

Поиск в глубину пройдет через узлы графа в последовательности ABDBEBACF. При этом типе поиска первоначально в каждой проходящей вершине выбирается крайняя левая дуга. Продвижение осуществляется до тех пор, пока не достигается или терминальный узел или цель. Если встретился терминальный узел, то необходимо вернуться на один уровень назад и свернуть направо, а затем продолжать держаться левой стороны, пока не встречается или цель, или новый терминальный узел. Если для какого-то промежуточного узла все исходящие дуги исследованы, и они ведут в тупик, то осуществляется возврат к предыдущему узлу. Этот процесс повторяется до тех пор, пока не будет найдена цель или не будут пройдены все узлы из области поиска. Поиск в глубину гарантированно находит цель (если она существует), потому что в самом худшем случае он переходит в полный перебор, который имел бы место, если бы вершина G была целью.

Если запустить программу (см. лабораторную работу), то должен быть получен следующий результат:

```
New York to Chicago to Denver to Los Angeles
distance is 3000
```

Из рис.11.6 видно, что это действительно первое решение, которое можно найти, используя поиск в глубину. В то же время оптимальное решение выглядит так: из Нью-Йорка в Торонто в Лос-Анджелес с расстоянием 2600 миль. Но полученное решение не такое уж плохое.

В рассмотренном примере методом поиска в глубину было найдено довольно хорошее решение при первой же попытке и без какого-либо возврата, причем хорошее решение. Однако, для достижения оптимального решения необходимо было бы пройти почти все узлы, что не так уж хорошо. Поиск в глубину может весьма слабо работать в тех ситуациях, когда необходимо исследовать особо сложную задачу, не имеющую решения. В этом случае, поиск в глубину будет тратить впустую значительное время.

Метод поиска в ширину

Метод поиска в ширину является противоположностью поиска в глубину. Поиск в ширину проверяет каждый узел на том же самом уровне прежде, чем он переходит к следующему уровню.

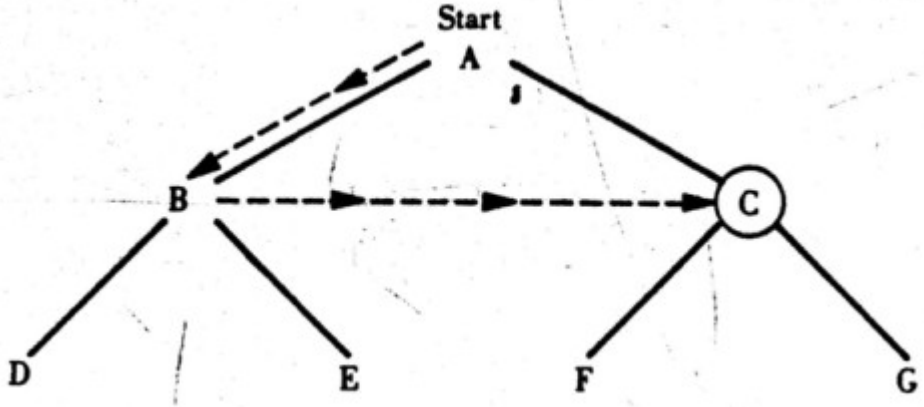

Рисунок 11.8. Путь решения (поиск в ширину, когда C является целью)

Рисунок показывает, что проход осуществляется через узлы ABC. Подобно поиску в глубину, поиск в ширину гарантирует обнаружение решения, если оно существует, потому что поиск в конечном счете вырождается в полный перебор.

Программы, реализующая поиск в ширину, дает следующее решение:

```
New York to Toronto to Los Angeles
distance is 2600
```

Оно является оптимальным решением.

Для данного примера поиск в ширину работает хорошо, без возвратов. Как оказалось, полученное решение является также оптимальным. Фактически, первые три решения, которые могут быть найдены, являются лучшими тремя маршрутами! Однако, нельзя обобщать этот результат на другие ситуации, потому что результат зависит от организации информации, записанной в компьютере. Однако, этот результат иллюстрирует радикальное различие поисков в глубину и в ширину.

Недостатки поиска в ширину проявляются, когда цель расположена на несколько уровней глубже. В этом случае, поиск в ширину прилагает существенное усилие, чтобы найти решение. Вообще, программист выбирает между поиском в глубину и поиском в ширину, делая предположение относительно того, где наиболее вероятно положение цели.

11.4.2. Эвристические методы поиска

Процедуры поиска в глубину и в ширину - "слепые". Они ищут решение, полагаясь исключительно на перемещение от одного узла к другому без использования каких-либо предположений, полученных на основе предварительного обучения. «Слепые» методы могут подходить для некоторых ситуаций, где программист располагает информацией, которая говорит, что необходимо использовать один метод или другой. Но программа, относящаяся к ИИ, требует поисковую процедуру, способную учитывать предыдущий опыт.

Эвристики *- это просто правила, которые позволяют сокращать поиск, указывая наиболее вероятно правильное направление*. Например, представим, что Вы потерялись в лесу и нуждаетесь в воде. Лес настолько густой, что Вы не можете видеть далеко вперед и деревья слишком высокие, чтобы залезть и оглянуться вокруг. Однако у Вас есть следующие знания: 1) реки, ручьи и водоемы наиболее вероятно располагаются в низинах; 2) животные часто вытаптывают тропы к местам водопоя; 3) когда Вы находитесь близко от воды, то можно почувствовать ее "запах"; 4) Вы можете услышать текущую воду. Так, в поисках пути к воде, Вы сначала спускаетесь вниз, потому что вода, вряд ли будет наверху. Затем, когда Вы натолкнулись на след оленей, который также ведет вниз, Вы следуете по нему, потому что Вы знаете, что он может вести к воде. Далее Вы слышите небольшой шум слева. Зная, что это может быть вода, Вы двигаетесь в этом направлении. Вы обнаруживаете усиленную влажность в воздухе. Вы можете почувствовать "запах" воды. Наконец, Вы находите ручей и утоляете жажду. Как показывает этот пример, эвристическая информация, хотя точно не гарантирует, но увеличивает вероятность того, что метод поиска найдет цель быстро или оптимально, или и то и то. Короче говоря, это увеличивает вероятность быстрого успеха.

Можно легко включить эвристическую информацию в программы, разрабатываемые для определенных применений, но уж очень сложно сделать обобщенный эвристический поиск.

Наиболее часто, эвристические методы поиска основаны на максимизировании или минимизировании некоторого аспекта задачи. Фактически, два эвристических подхода, которые далее рассматриваются, используют противоположные эвристики и выдают различные результаты. В качестве основы (базы) оба этих подхода используют процедуру поиска в глубину.

Метод поиска "восхождением на холм"

В задаче планирования полета из Нью-Йорка в Лос-Анджелес, есть два параметра, которые возможно пассажир захочет минимизировать. Первый - число необходимых пересадок. Второй - длина маршрута. Причем, самый короткий маршрут не обязательно подразумевает наименьшее количество пересадок. Алгоритм поиска, который ищет решение, минимизирующее количество пересадок, будет использовать эвристику: «чем большее расстояние покрывается за каждый перелет, тем выше вероятность того, что пассажир оказывается ближе к месту назначения и, таким образом, сокращается число пересадок». В ИИ, этот тип метода поиска называется "восхождение на холм".

Формально, алгоритм "восхождения на холм", выбирает в качестве следующего шага узел, который, кажется, что размещен наиболее близко к цели (т.е. смежный узел наиболее удаленный от текущего). Алгоритм получил название из аналогии с путешественником, который потерялся в темноте на полпути к вершине горы. Если путешественник предполагает, что лагерь туристов находится на горе, то, даже в темноте, путешественник знает, что каждый шаг, который он сделает вверх - это шаг в правильном направлении!

Для задачи об авиарейсах можно включить в программу следующую эвристику для алгоритма "восхождения на холм": выбрать смежный рейс, место назначения которого является насколько возможно дальше от текущего положения, в надежде, что это будет ближе к конечной цели.

Для контрольного примера эта программа получит решение:

New York to Denver to Los Angeles
distance is 2900

Этот результат весьма хорош! Маршрут имеет минимальное число остановок на пути - только одну и весьма близкую к минимальной протяженность маршрута. Кроме того, программа не тратит впустую время или усилия на обширный возврат.

Однако, если не существует связи Денвер–Лос-Анджелес, то поиск не обеспечит хороший результат. В этом случае решение было бы: Нью-Йорк - Денвер - Хьюстон - Лос-Анджелес, которое имеет протяженность 4900 миль! Это решение предполагает восхождение на ложный холм, потому что перелет в Хьюстон не приближает к цели - Лос-Анджесу.

Во многих случаях поиск по методу "восхождения на холм" довольно хорош, потому что он может уменьшать число узлов, через которые необходимо пройти в процессе получения решения. Однако, он может потерпеть неудачу из-за наличия трех следующих **недостатков**. Во-первых, - это существование ложных холмов, как показано на примере. Во-вторых, - это возможное наличие "плато", на котором все последующие шаги на первый взгляд одинаково хороши (или плохи). В этом случае поиск "восхождением на холм" не лучше, чем поиск в глубину. Третья проблема - наличие "горных хребтов". В этом случае алгоритм поиска "восхождением на холм" не эффективен, потому что он пересечет горный хребет несколько раз в процессе возвратов.

Несмотря на эти потенциальные неприятности, поиск "восхождением на холм" в основном ведет к хорошему решению быстрее, чем любой неэвристический метод.

Поиск по минимальной стоимости

Противоположностью "восхождению на холм" является поиск по минимальной стоимости. Эта стратегия подобна той ситуации, когда, находясь на середине улицы на большом холме, вы одеваете роликовые коньки, и у вас определенно есть ощущение, что намного легче ехать вниз, чем подниматься наверх! Таким образом, поиск по минимальной стоимости выберет путь, предполагающий наименьшее количество усилий.

Применительно к задаче об авиарейсах поиск по минимальной стоимости подразумевает, что программа будет брать самый короткий рейс во всех случаях так, чтобы найденный маршрут имел хороший шанс быть самым коротким. В отличие от поиска "восхождением на холм", который минимизирует, число пересадок, поиск по минимальной стоимости минимизирует протяженность маршрута.

Решение получается следующим:

```
New York to Toronto to Los Angeles
distance is 2600
```

Это показывает, что в данном примере поиск по минимальной стоимости находит самый короткий маршрут.

Поиск по минимальной стоимости имеет те же самые преимущества и **недостатки** что и "восхождение на холм". Могут встречаться ложные долины, низменности, и ущелья; но в целом, поиск по минимальной стоимости имеет тенденцию работать довольно хорошо. Однако, нельзя утверждать, что поиск по минимальной стоимости вообще лучше только потому, что он работает лучше "восхождения на холм" на этой специфической задаче. Здесь можно только сказать, что в среднем он выиграет у "слепого" поиска.

Выбор метода поиска

Эвристические методы имеют тенденцию работать лучше, чем "слепой" поиск. Однако, не всегда возможно использовать эвристический поиск, потому что можно не располагать достаточной информацией, позволяющей определить следующий узел, наиболее вероятно приближающий к цели. Поэтому задачи можно разделить на две категории: 1) которые могут использовать эвристический поиск, 2) которые не могут.

Если при решении задачи нельзя применить эвристику, то лучше выбрать поиск в глубину, потому что он будет работать в основном лучше. Единственное исключение из этого правила - это ситуация, когда известно, что поиск в ширину будет работать лучше.

Выбор между "восхождением на холм" и поиском по минимальной стоимости зависит от налагаемых ограничений. Поиск "восхождением на холм" вырабатывает решение с наименьшим числом промежуточных узлов, а поиск по минимальной стоимости находит путь, который требует наименьших усилий.

Если необходимо получать почти оптимальное решение, но нет возможности применить исчерпывающий поиск (полный перебор), то следует попробовать каждую из технологий поиска и далее использовать наиболее подходящую. Это правило эффективно, потому что работа различных видов поиска существенно различна на различных задачах.

11.4.3. Получение множественных решений

Иногда приходится находить несколько решений той же самой проблемы. Однако, это не та ситуация, в которой необходимо найти все решения, как это происходит при исчерпывающем поиске. Например, вообразите, что вам проектируют "дом вашей мечты": Архитектор делает набросок нескольких различных планов зданий, соответствующих вашим запросам. Он не делает эскизы всех возможных вариантов зданий, которые в принципе вам подошли бы. Их может быть очень много. Небольшой набор эскизов зданий, которые вам понравились бы, и составляет в этом случае множественное решение. Множественное решение представляет собой набор вариантов, из которого можно выбрать наиболее подходящий. Множественное решение – это подмножество множества допустимых решений.

Существует много методов выработки множественных решений, здесь рассмотрим только два: 1) удаление пути; 2) удаление узла. Как подразумевают названия этих методов, выработка множественных решений требует удаления из системы уже найденных решений. Ни один из этих методов не используется для нахождения всех решений. Обнаружение всех решений - это другая задача и в основном требует исчерпывающего поиска.

Метод с удалением пути

Метод удаления пути вырабатывает множество решений, удаляя из базы данных все узлы, которые формируют текущее решение, и затем пытается найти другое решение. В сущности, метод удаления пути сокращает количество ветвей дерева.

Программа с этой версией main() найдет следующие решения:

```
New York to Chicago to Denver to Los Angeles
distance is 3000
New York to Toronto to Los Angeles
distance is 2600
New York to Denver to Los Angeles
distance is 2900
```

Интересно, что найдены три лучших решения. Однако, нельзя делать обобщения на основе этого результата, потому что он основан на способе, которым данные помещены в базу данных и на конкретной ситуации.

Метод с удалением узла

Метод с удалением узла - другой способ выработки множественных решений. Он просто удаляет последний узел для текущего решения и затем пробует искать снова.

Программа получит следующий результат:

```
New York to Chicago to Denver to Los Angeles
distance is 3000
New York to Chicago to  Denver  to  Houston  to  Los  Angeles
distance is 5000
New York to Toronto to Los Angeles
distance is 2600
```

Второе решение содержит самый плохой из возможных маршрутов, но программа находит и оптимальное решение. Помните, что эти результаты невозможно обобщить, потому что они основаны на физической организации данных в базе данных и на определенной изучаемой задаче.

Нахождение оптимального решения

Все описанные методы поиска ориентированны на обнаружение решения. Как показывает эвристический поиск, были применены усилия, чтобы увеличить вероятность обнаружения "хорошего" и, предпочтительно лучшего, решения. Однако, иногда желательно только оптимальное решение. Термин "оптимальное решение" используется здесь для обозначения наилучшего маршрута, который можно найти, используя одну из различных технологий нахождения множественных решений. Фактически оно может и не быть лучшим решением. (Обнаружение действительно оптимального решения требовало бы предельно дорогого исчерпывающего поиска.)

Для получения «оптимального» решения первоначально формируется множественное решение. Оно представляет собой множество альтернатив, из которых выбирается наилучшая. Поскольку множественное решение является подобластью области допустимых решений, то полученное «оптимальное» решение может быть не самым лучшим, т.е. самое лучшее решение может находиться вне множественного решения.