Лекция № 11

Методы поиска. Часть 1

Search methods. Part 1

МЕТОДЫ ПОИСКА РЕШЕНИЙ

ДВА ПОДХОДА К МОДЕЛИРОВАНИЮ МЫШЛЕНИЯ

1. **Символический подход.** Исторически был доминирующим подходом к моделированию интеллекта. Согласно этому подходу все интеллектуальные действия сводятся к оперированию символами или понятиями. Он *включает построение формальных моделей и соответствующих им механизмов рассуждений*.

2. **Нейро-сетевой или нейрокибернетический подход.** Является противоположностью символического подхода к исследованию и моделированию интеллекта. Основная идея этого направления: единственный объект, способный мыслить, — это человеческий мозг. Поэтому любое мыслящее устройство должно каким-то образом воспроизводить его структуру.

Это подход ориентирован на *программно-аппаратное моделирование структур, подобных структуре мозга.* Делается упор на создание элементов, аналогичных нейронам, и их объединение в функционирующие системы, т.е. в нейронные сети (НС).

МЕТОДЫ ПОИСКА РЕШЕНИЙ

1. Символические системы и поиск

Анализируя работы в области искусственного интеллекта, «отцы ИИ» - Аллен Ньюэлл и Герберт Саймон - выделили два основных понятия: *символические системы* и *поиск*.

Символическая система есть набор символов, образующих символические структуры, и набор процессов.

Символ есть первичное понятие.
Примером символа является строка буквенно-цифровых знаков (*список, изделие-35, 3.14* и т.п.).

Символическая структура представляет собой совокупность символов, соотнесенных определенным физическим способом (например, один символ следует за другим).

МЕТОДЫ ПОИСКА РЕШЕНИЙ

Примером символических структур являются следующие наборы символов: (ИДЕНТИЧНОСТЬ ОБЪЕКТ-1 ОБЪЕКТ-2); (ПЛЮС *X* У); (ЕСТЬ (НИКОЛАЙ) (НАЧАЛЬНИК (ОТДЕЛ 237))).

Процессы способны производить, разрушать и модифицировать символические структуры.

Символическая структура может рассматриваться как *тип данных* в некотором языке.

Символические структуры обладают ***двумя основными свойствами***:
1) они могут обозначать (designate) объекты, процессы и другие символические структуры;
2) если они обозначают процессы, то они могут быть интерпретированы.

МЕТОДЫ ПОИСКА РЕШЕНИЙ

Символическая структура ***обозначает*** некоторую сущность (объект, процесс или символическую структуру), если символическая система может осуществлять поведение, определяемое данной сущностью, или может воздействовать на эту сущность.

Система может ***интерпретировать*** символическую структуру, если структура обозначает некоторый процесс, и система может выполнить этот процесс.

МЕТОДЫ ПОИСКА РЕШЕНИЙ

Ньюэлл и Саймон в 1976 году высказали *две гипотезы*, на которых базируются исследования по искусственному интеллекту:

1. Гипотеза символических систем. (Физические) символические системы имеют необходимые и достаточные средства для осуществления интеллектуальных действий в широком смысле.

2. Гипотеза поиска. Решения задач могут быть представлены в виде символических структур. Символические системы решают задачи с помощью поиска, т.е. они генерируют потенциальные решения и постепенно модифицируют их, пока они не будут удовлетворять условиям решения**.**

Приведенные гипотезы разделяются многими специалистами в области искусственного интеллекта. По-видимому, можно утверждать, что все существующие интеллектуальные системы (например, «Общий решатель задач», созданный Ньюэллом и Саймоном, экспертные системы) подтверждают эти гипотезы.

МЕТОДЫ ПОИСКА РЕШЕНИЙ

2. Классификация методов поиска решений

Методы решения задач, основанные на сведении их к поиску, зависят от особенностей проблемной области, в которой решается задача, и от требований (ограничений), предъявляемых пользователем к решению. Эти особенности и требования определяют сложность задачи.

Сложность решаемой задачи характеризуется следующим набором параметров:

1) размер пространства, в котором решается задача;
2) изменяемость области;
3) полнота модели;
4) определенность данных;
5) количество необходимых решений;
6) ограничения на результат и способ его получения.

МЕТОДЫ ПОИСКА РЕШЕНИЙ

Параметр ***"изменяемость области"*** характеризует степень изменяемости области во времени и пространстве. По параметру *"изменяемость"* выделяются *статические* и *динамические области*.

Параметр ***"полнота модели"*** характеризует адекватность модели, используемой для описания данной проблемной области. Обычно, если модель не полна, то для описания области используют несколько моделей, дополняющих друг друга за счет отражения различных свойств проблемной области (например, модель, описывающая электрическую схему двигателя, и модель, определяющая его механическую схему).

МЕТОДЫ ПОИСКА РЕШЕНИЙ

Параметр ***"определенность данных"*** характеризует степень *точности/ошибочности* и *полноты/неполноты* данных.

"Точность" является показателем того, что проблемная область с точки зрения решаемых задач описана точными данными.

"Ошибочность" является показателем того, что данные о проблемной области не точны.

Под ***"полнотой"*** (неполнотой) данных понимается достаточность (недостаточность) входных данных для однозначного решения задачи. Обычно при неполноте данных для поиска решения необходимо использовать некоторые предположения или ограничения.

МЕТОДЫ ПОИСКА РЕШЕНИЙ

Требования пользователя к результату задачи, решаемой с помощью поиска, можно характеризовать параметрами *"количество решений"* и *"ограничения на результат и способ его получения"*.

Параметр ***"количество решений"*** может принимать следующие основные значения: *одно решение, несколько решений, все решения*.

Параметр ***"ограничения на результат и способ его получения"*** определяют определенные требования к решению.

Так, например, для системы, выдающей рекомендации по лечению больных, пользователь может указать требование не использовать некоторое лекарство (в связи с его отсутствием или в связи с тем, что оно противопоказано данному пациенту).

Этот параметр может определять и такие особенности, как *время решения* ("не более чем"), *объем памяти*, используемой для получения результата, *указание об обязательности* (невозможности) *использования каких-либо знаний* (данных) и т.п.

МЕТОДЫ ПОИСКА РЕШЕНИЙ

Во введенных терминах сложность задачи колеблется от **простых задач** (малая область, статическая область, полная модель, определенные данные, какое-нибудь решение, отсутствие ограничений на результат и способ его получения)

до **сложных задач** (большая область, динамическая область, неполнота одной модели, ошибочные и неполные данные, все решения, произвольные ограничения на результат и способ его получения).

МЕТОДЫ ПОИСКА РЕШЕНИЙ

Классификация методов решения задач на основе поиска:

- **Методы поиска в одном пространстве** – методы, предназначенные для использования в следующих условиях: малые области, статические области, полнота модели, точные и полные данные.
- **Методы поиска в иерархических пространствах** – методы, предназначенные для работы в больших статических областях.
- **Методы поиска в динамической проблемной области** – методы, предназначенные для работы с областями, изменяемыми во времени и/или в пространстве.
- **Методы поиска при неточных и неполных данных**.
- **Методы поиска, использующие несколько моделей**, – методы, предназначенные для работы с областями, для адекватного описания которых одной модели недостаточно.

Поиск решения в одном пространстве

Поиск в пространстве состояний.

Задача поиска в пространстве состояний может быть сформулирована следующим образом.

Пусть задана тройка (S_0, F, S_T),

где S_0 – множество начальных состояний (условия задачи),

F – множество операторов задачи, отображающих одни состояния в другие,

S_T – множество конечных (целевых) состояний (решений задачи).

В этой постановке решить задачу – значит **определить такую последовательность операторов,** которая **преобразует начальные состояния в конечные**.

Поиск решения в одном пространстве

Процесс решения можно представить в виде графа $G = (X, Y)$, где

$X = \{ x_0, x_1, \ldots \}$ − множество (в общем случае бесконечное) вершин графа, каждая из которых отождествляется с одним из состояний,

Y − множество (дуг), содержащее пары вершин (x_i, x_j), $x_i, x_j \in X$.

Поиск в пространстве состояний естественно представлять в виде ориентированного графа. Наличие пары (x_i, x_j) свидетельствует о существовании некоторого оператора f $(f \in F)$, преобразующего состояние, соответствующее вершине x_i в состояние x_j.

Поиск решения в одном пространстве

С точки зрения поиска в пространстве состояний для некоторой вершины x_i уместно выделить множество всех направленных пар $(x_i, x_j) \in Y$, т.е. множество дуг, исходящих из вершины x_i (родительской вершины), и множество вершин (называемых дочерними вершинами), в которые эти дуги приводят.

Множество дуг, исходящих из вершины x_i, соответствует множеству операторов, которые могут быть применены к состоянию, соответствующему вершине x_i.

В множестве вершин X выделяют подмножество вершин $X_0 \subseteq X$, соответствующее множеству начальных состояний (S_0) и подмножество вершин $X_T \subseteq X$, соответствующее множеству конечных (целевых) состояний (S_T).

Множество X_T может быть задано как явно, так и неявно, т.е. через свойства, которыми должны обладать целевые состояния.

Поиск решения в одном пространстве

Определим на графе G **маршрут** L как такую последовательность ребер, что каждые два соседних ребра имеют общую концевую точку.

В случае ориентированного графа маршрут называют *путем*.

Будем обозначать **путь** L как последовательность $(x_{i1}, x_{i2},\ldots, x_{i(k+1)})$, где пара $(x_{ij}, x_{i(j-1)}) \in Y, j = 2, \ldots, k + 1$.

В этом случае путь L имеет длину k и соединяет вершины (состояния) x_{i1} и $x_{i(k+1)}$.

Очевидно, что решение задачи методом поиска в пространстве состояний (т.е. нахождение последовательности операторов, преобразующей начальное состояние в конечное) сводится к задаче поиска пути L на графе G.

Путь из $x_0 \in X_0$ в $x_T \in X_T$ называют **решающим** (целевым).

Поиск решения в одном пространстве

Часто бывает удобно приписывать дугам графа веса, отражающие *стоимость* применения соответствующих операторов.

Для обозначения стоимости дуги, направленной из x_i в x_j, используют запись $c(x_i, x_j)$.

Стоимость пути между двумя вершинами определяется как сумма стоимостей всех дуг, образующих этот путь.

В ряде приложений возникает задача нахождения путей (пути), имеющих минимальную стоимость, между любыми элементами из множества X_0 и любыми элементами из множества X_T.

Поиск решения в одном пространстве

Граф G может быть задан как явно, так и неявно.

Неявное задание графа G состоит в определении множества X_0 и множества операторов F, которые будучи применимы к некоторой вершине графа, дают все ее дочерние вершины.

Граф G задает **пространство состояний**, т.е. пространство, **в котором осуществляется поиск решения.**

Построение пространства осуществляется с помощью следующего процесса. Берется некая вершина из $x_0 \in X_0$, к ней применяются все возможные операторы, порождающие все дочерние вершины.

Порождение всех дочерних вершин для некоторой вершины x_i называют процессом **раскрытия вершин**. Если получена целевая вершина, то она не раскрывается.

Поиск решения в одном пространстве

Процесс построения пространства состояний (раскрытия вершин) заканчивается, когда все нераскрытые вершины являются **целевыми** или **терминальными** (т.е. вершинами, к которым нельзя применить никаких операторов).

От каждой дочереней вершины к породившей ее устанавливаются **указатели**. Эти указатели позволяют найти путь назад к начальной вершине, после того, как обнаружена целевая вершина. После обнаружения целевой вершины по этим указателям выявляется **путь решения**. Операторы над описаниями состояний, связанные с дугами этого пути, образуют **решающую последовательность**.

Поиск решения в одном пространстве

В связи с тем, что пространство состояний может содержать бесконечное количество вершин, на практике процесс порождения пространства *ограничивают* либо *временем*, либо *объемом памяти*.

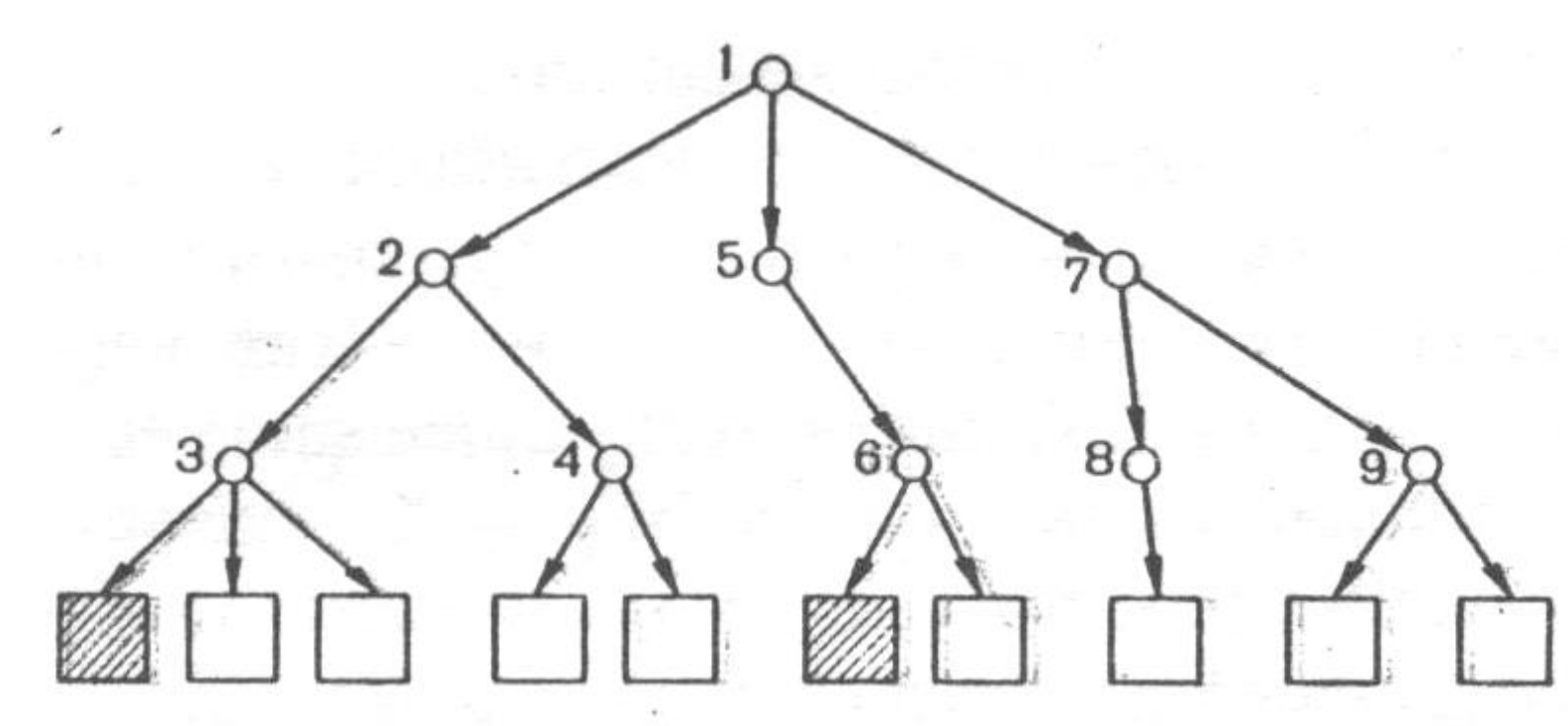
В практических приложениях часто требуется обеспечить полноту поиска, т.е. организовать поиск так, чтобы все целевые вершины были найдены, если они существуют.

Надежным способом обеспечения полноты является **полный перебор всех вершин**.

Для задания процесса перебора необходимо определить порядок, в котором будут перебираться вершины графа. Обычно выделяют два основных способа поиска: ***поиск в глубину*** и ***поиск в ширину***.

Поиск в глубину

При поиске в глубину сначала раскрывается та вершина, которая была построена самой последней.



Замечание. Вершины пронумерованы в том порядке, в котором они раскрываются (не порождаются), целевые вершины помечены заштрихованными квадратами, а терминальные — белыми квадратами.

Поиск в глубину

Глубина вершины в графе определяется следующим образом:

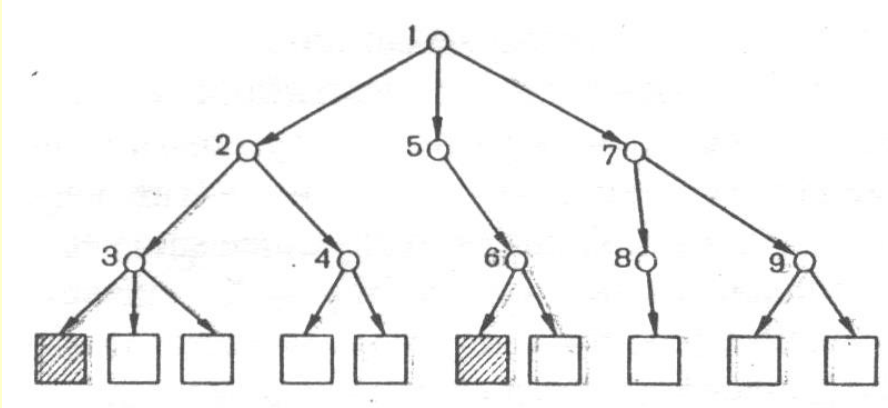
1) глубина начальной вершины равна нулю;
2) глубина неначальной вершины равна единице плюс глубина наиболее близкой родительской вершины.

При практической реализации поиск в глубину в некотором направлении завершается в следующих случаях:

1) при достижении целевой вершины;
2) при достижении терминальной вершины;
3) при построении в ходе поиска вершины, глубина которой превышает некоторую граничную глубину.

Поиск в глубину

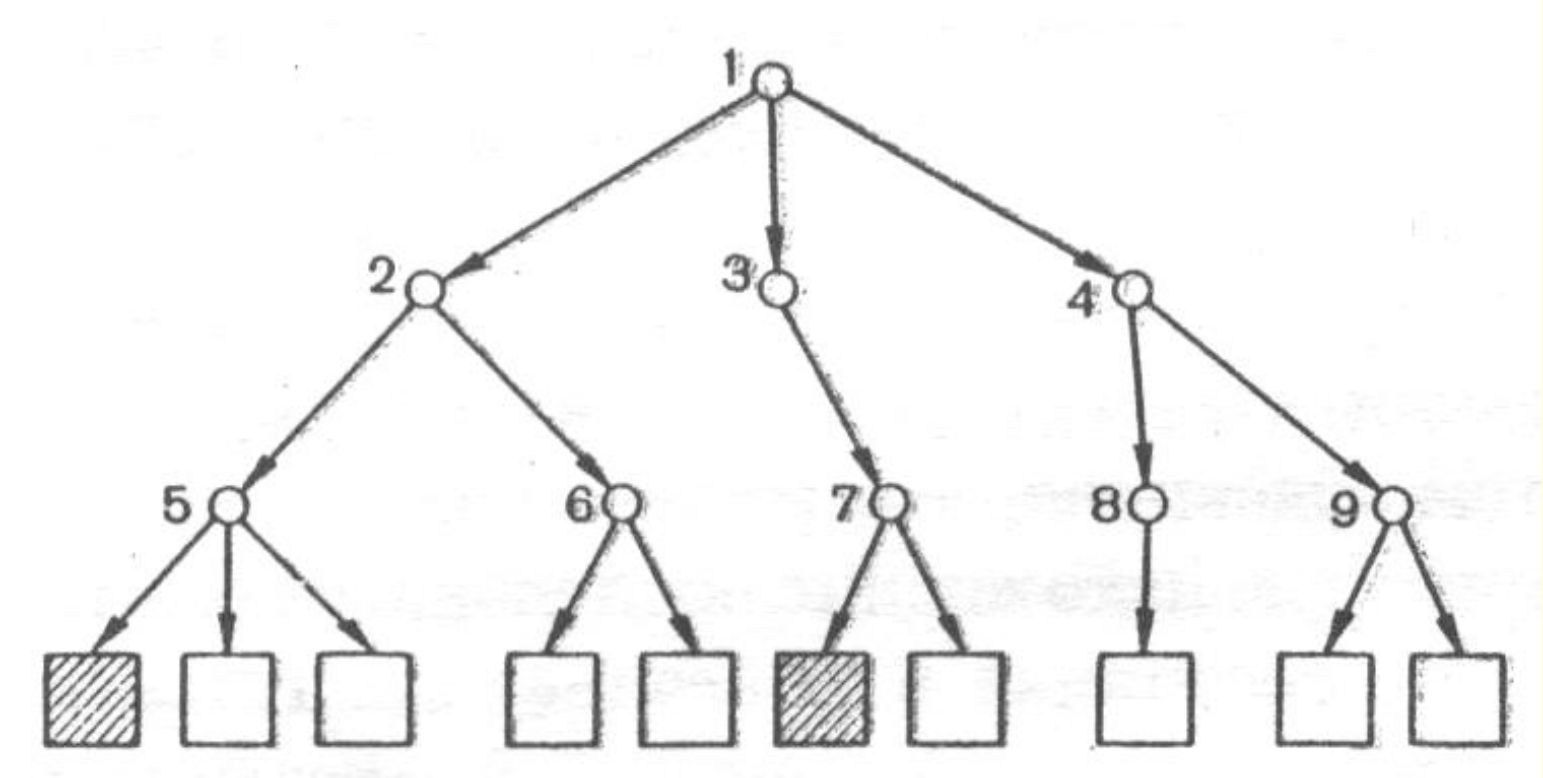
Алгоритм перебора в глубину:



1) Поместить начальную вершину в список ОТКРЫТ.
2) Если ОТКРЫТ пуст, то НЕУДАЧА, иначе перейти к шагу 3.
3) Взять первую вершину *n* из списка ОТКРЫТ и перенести ее в список ЗАКРЫТ.
4) Если глубина вершины *n* равна граничной глубине, то переход к шагу 2, иначе к шагу 5.
5) Раскрыть вершину *n*, поместить все ее дочерние вершины в ***начало*** списка ОТКРЫТ и построить указатели, ведущие от них к *n*. Если дочерних вершин нет, то перейти к шагу 2.
6) Если среди дочерних вершин есть целевые, то выдать решение; в противном случае перейти к шагу 2.

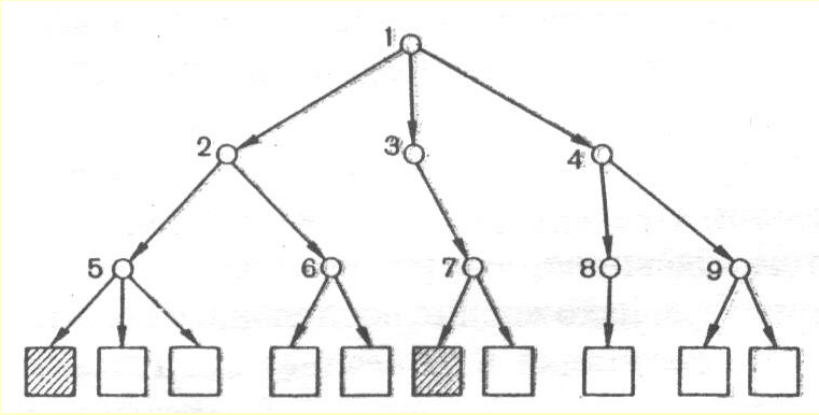
Поиск в ширину

При поиске в ширину вершины раскрываются в том же порядке, в котором они порождаются [*Нильсон Н. Искусственный интеллект. Методы поиска решений. М.: Мир, 1973*.]



Поиск в ширину

Алгоритм поиска в ширину (простой метод полного перебора):



1) Поместить начальную вершину в список ОТКРЫТ.
2) Если список ОТКРЫТ пуст, то выдать НЕУДАЧА, иначе перейти к шагу 3.
3) Взять первую вершину *n* из списка ОТКРЫТ и перенести ее в список ЗАКРЫТ.
4) Раскрыть вершину *n*, поместить все ее дочерние вершины в ***конец*** списка ОТКРЫТ и построить указатели, ведущие от них к *n*. Если дочерних вершин нет, то перейти к шагу 2.
5) Если среди дочерних вершин есть целевые, то выдать решение; в противном случае перейти к шагу 2.

Прямой и обратный поиск

При использовании каждого из методов могут быть найдены все решения. При переборе всего пространства оба метода будут анализировать одинаковое количество вершин, однако метод поиска в ширину будет требовать существенно больше памяти, так как он запоминает все пути поиска (а не один, как при поиске в глубину).

Если в пространстве состояний ввести операторы (F^{-1}), переводящие состояние S_t в предшествующее состояние S_{t-1}, то поиск можно осуществлять не только в направлении от начального состояния к целевому, но и в обратном направлении.

Поиск первого типа называют поиском от данных (или ***прямым поиском***), а поиск второго типа — поиском от цели (или ***обратным поиском***).

Можно организовать поиск в обоих направлениях одновременно. Такой поиск называют ***двунаправленным*** (или бинаправленным).

Метод равных цен

Любым из методов полного перебора непременно будет найден самый короткий путь к целевой вершине, если он существует.

Однако есть задачи, которые предъявляют различные требования к решению. Например, требуется найти решение, имеющее минимальную стоимость. В этом случае задается функция стоимости $c(n_i, n_j)$, задающая стоимость перехода из вершины n_i в вершину n_j.

Метод равных цен позволяет найти путь, **стоимость** которого **минимальна**.

В методе равных цен для каждой вершины n в дереве перебора нужно помнить стоимость пути, построенного от начальной вершины s к вершине n. Пусть $g^(n)$ – стоимость пути от s к n. В случае деревьев перебора $g^(n)$ можно быть уверенным, что это стоимость пути, имеющего *минимальную стоимость*.

В методе равных цен **вершины раскрываются в порядке возрастания стоимости** $g^(n)$.

Метод равных цен

Метод равных цен:

1) Поместить начальную вершину *s* в список ОТКРЫТ. Положить *g^(s) = 0*,
2) Если список ОТКРЫТ пуст, то НЕУДАЧА, иначе перейти к 3.
3) Взять из списка ОТКРЫТ ту вершину (*n)*, для которой величина *g^(n)* имеет наименьшее значение, и поместить ее в список ЗАКРЫТ. (При выборе целевая вершина имеет приоритет.)
4) Если *n* целевая вершина, то выдать решение, иначе переход к шагу 5.
5) Раскрыть вершину *n*, построив все ее дочерние вершины. Если дочерних вершин нет, то перейти к шагу 2.
   Для каждой дочерней вершины n_i вычислить стоимость $g^(n_i)$, положив $g^(n_i) = g^(n) + c(n, n_i)$. Поместить эти дочерние вершины вместе с вычисленными стоимостями в список ОТКРЫТ и построить указатели, идущие назад к *n*.
6) Перейти к шагу 2.

Метод равных цен

Метод равных цен можно использовать для поиска минимального пути, если положить *стоимость каждого ребра равной 1*.

Если имеется несколько начальных вершин, то алгоритм модифицируется на шаге (1): все начальные вершины помещаются в список ОТКРЫТ.

Если целевые состояния могут быть описаны явно, то процесс перебора можно пустить в обратном направлении, приняв целевые вершины в качестве начальных и используя обращение операторов *F*.

Поиск решения в одном пространстве. Итоги

Перебор на произвольных графах

При поиске на графах, а не на деревьях, нужно внести небольшие изменения в описанные алгоритмы.

В простом методе полного перебора (поиск в ширину) нужно проверять, не находится ли уже вновь построенная вершина в списках ОТКРЫТ и ЗАКРЫТ: если вершина уже строилась раньше ее не нужно вновь помещать в список ОТКРЫТ.

В алгоритме перебора в глубину нужно выбирать для раскрытия самую глубокую вершину из списка ОТКРЫТ (без превышения граничной глубины). Если порождается вершина, уже имеющаяся либо в списке ОТКРЫТ, либо ЗАКРЫТ, необходимо пересчитать глубину такой вершины.

Заметим, что даже при поиске на полном графе построенное множество вершин и указателей тем не менее образует *дерево*.

Эвристический поиск

Методы поиска в глубину и ширину называют *слепым поиском*, поскольку в этих методах порядок раскрытия вершин предопределен и никак не зависит от расположения цели.

При увеличении пространства поиска методы слепого поиска требуют чрезмерных затрат времени и (или) памяти. Стремление сократить время поиска привело к созданию **эвристических методов поиска**, т.е. методов, использующих некоторую (эвристическую) информацию о проблемной области для рассмотрения не всего пространства поиска, а таких путей в нем, которые с наибольшей вероятностью приводят к цели.

Эвристический поиск

Один из путей уменьшить перебор состоит в выборе более информированного оператора *F*, который не строит не относящихся к делу вершин.

Другой путь – использовать эвристическую информацию для модификации шага (5) алгоритма перебора в глубину: вновь построенные вершины располагать не в произвольном порядке, а в порядке, зависящем от эвристической информации. Тогда при переборе в глубину в первую очередь будет раскрываться вершина, которая представляется наилучшей.

Более гибкий (и дорогой) путь использования эвристической информации состоит в том, чтобы на каждом шаге переупорядочивать вершины списка ОТКРЫТ.

Эвристический поиск

Таким образом, хорошим способом сокращения перебора является использование эвристической информации для определения на каждом шаге дальнейшего *направления перебора*.

Для этого необходимо ввести меру "***перспективности***" вершины в виде некоторой ***оценочной функции***.

Как правило, оценочные функции пытаются количественно оценить расстояние от текущей вершины до конечной. Из двух вершин при одинаковой глубине считается более перспективной та, от которой меньше расстояние до цели.

В некоторых случаях удается ввести такую оценочную функцию, что она, сокращая перебор, не теряет свойства полноты. Чаще же используемые эвристики, сильно сокращая перебор, влекут за собой потерю свойства полноты.

Эвристический поиск

Использование оценочных функций

Оценочная функция должна обеспечивать возможность ранжирования вершин – кандидатов на раскрытие – с тем, чтобы выделить ту вершину, которая с наибольшей вероятностью находится на лучшем пути к цели.

Пусть *f* – оценочная функция, а *f(n)* – значение этой функции на вершине *n*.

Расположим вершины, предназначенные для раскрытия, в порядке *возрастания* их значений функции *f*.

Тогда можно построить *алгоритм упорядоченного перебора* (ordered-search algorithm), который для очередного раскрытия выбирает ту вершину из списка ОТКРЫТ, для которой значение *f* оказывается наименьшим.

Эвристический поиск (*Алгоритм упорядоченного перебора*)

1) Поместить начальную вершину s в список ОТКРЫТ и вычислить $f(s)$.
2) Если список ОТКРЫТ пуст, то НЕУДАЧА, иначе перейти к 3.
3) Взять из списка ОТКРЫТ ту вершину (n), для которой величина f имеет наименьшее значение, и поместить ее в список ЗАКРЫТ.
4) Если n целевая вершина, то выдать решение, иначе переход к шагу 5.
5) Раскрыть вершину n, построив все ее дочерние вершины. (Если дочерних вершин нет, то перейти к шагу 2.) Для каждой дочерней вершины n_i вычислить значение $f(n_i)$.
6) Связать с теми из вершин n_i, которых еще нет в списках ОТКРЫТ и ЗАКРЫТ только что посчитанные значения $f(n_i)$. Поместить эти вершины в список ОТКРЫТ и провести от них к вершине n указатели.
7) Связать с теми дочерними вершинами n, которые уже были в списках ОТКРЫТ и ЗАКРЫТ, меньшее из прежних и только что вычисленных значений f. Поместить в список ОТКРЫТ те дочерние вершины, для которых новое значение f оказалось ниже, и изменить направление указателей от всех вершин, для которых значение f уменьшилось, направив их к n.
8) Перейти к шагу 2.

Эвристический поиск

Алгоритм A* (A star)

Алгоритм A* – это алгоритм упорядоченного перебора, в котором используется оценочная функция ***f(n)*** вида:

$f(n) = g(n) + h(n)$,

где ***g(n)*** – действительная стоимость оптимального пути (расстояние) от начальной вершины ***s*** до вершины ***n***,

h(n) – стоимость оптимального пути (расстояние) от вершины n до целевой вершины.

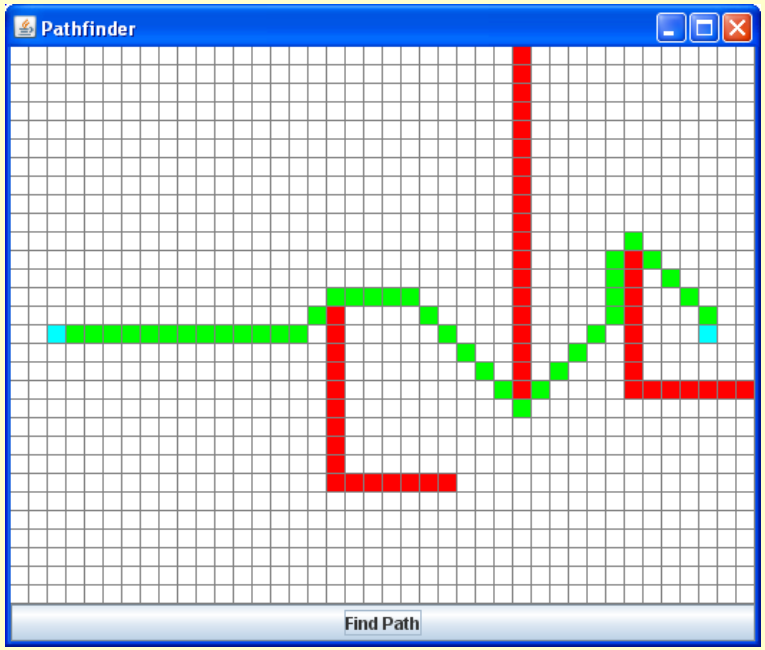
Значение ***f(n)*** есть стоимость оптимального пути при условии, что он проходит через вершину ***n***.

Заметим, что $f(s) = h(s)$ представляет собой действительную стоимость оптимального пути от начальной вершины s к цели без всяких ограничений.

Эвристический поиск

Алгоритм A* (A star)

Функция ***h(n)*** должна быть **допустимой эвристической оценкой**, то есть не должна переоценивать расстояния к целевой вершине. Например, для задачи маршрутизации ***h(n)*** может представлять собой расстояние до цели по прямой линии, так как это физически наименьшее возможное расстояние между двумя точками.



Эвристический поиск

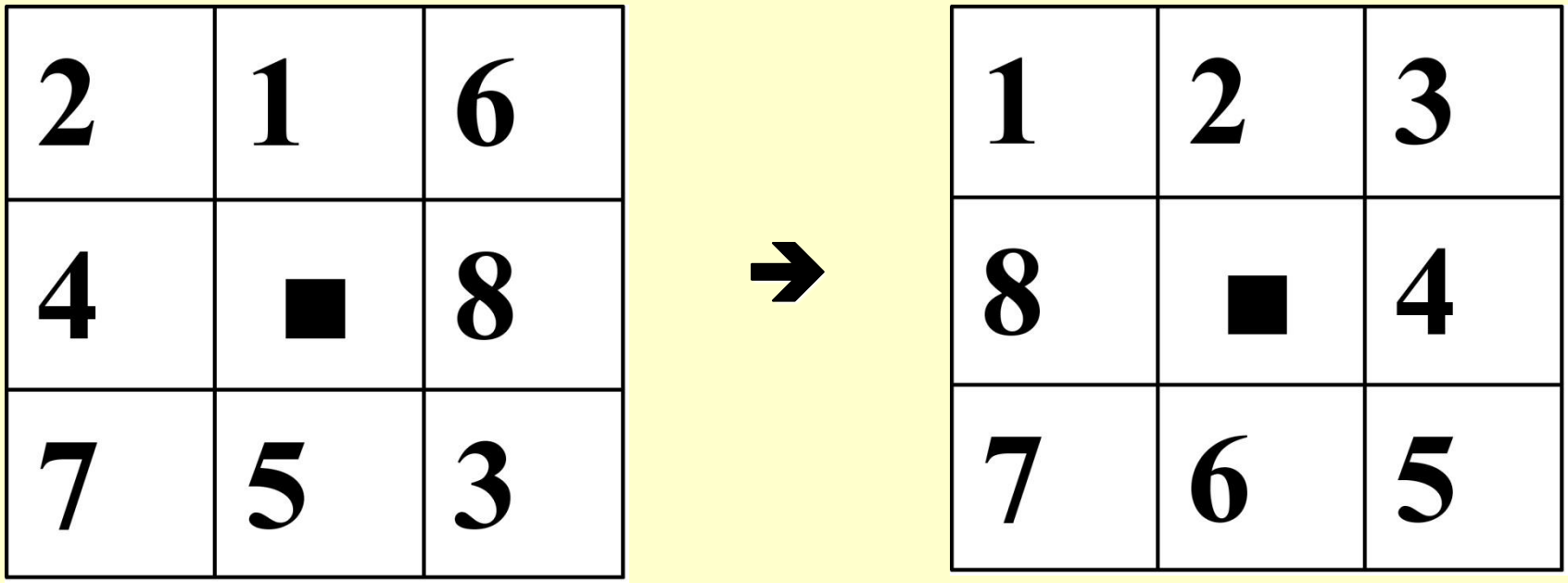
Алгоритм A* (A star)

При использовании A* для моделирования перемещения по поверхности, покрытой координатной сеткой, можно использовать следующие функции $h(n)$:

- Если перемещение возможно в четырех направлениях, то в качестве эвристики часто выбирается манхэттенское расстояние[1] $h(n)=|n.x-goal.x|+|n.y-goal.y|$.
- Когда к четырем направлениям добавляются диагонали выбирается расстояние Чебышёва: $h(n)=max(|n.x-goal.x|,|n.y-goal.y|)$.
- Если передвижение не ограничено сеткой, то можно использовать евклидово расстояние по прямой: $h(n)=sqrt((n.x-goal.x)^2+(n.y-goal.y)^2)$.

Эвристический поиск

Игра в восемь



Эвристический поиск

Для игры в восемь может быть использована оценочная функция ***h(n) = W(n),***
где ***W(n)*** —число фишек, находящихся не на своих местах.

Но эта функция *не обеспечивает хорошей оценки* трудности данного расположения фишек (в смысле числа шагов, отделяющих от цели).

Лучшую оценку дает функция ***h(n) = P(n),***
где ***P(n)*** — сумма расстояний каждой фишки от «своего места» (без учета фишек, расположенных на ее пути).

Однако даже *эта оценка слишком груба*, так как в ней не учитывается трудность обмена местами двух соседних фишек.

2	1	6
4	■	8
7	5	3

1	2	3
8	■	4
7	6	5

Эвристический поиск

Следующая функция дает еще лучшую оценку:

$$h(n) = P(n) + 3S(n),$$

здесь

P(n) — по-прежнему сумма расстояний каждой фишки от «своего места»;

S(n) — число очков, учитывающее порядок расположения фишек.

Для вычисления ***S(n)*** нужно последовательно просмотреть все нецентральные фишки в данной конфигурации и за каждую фишку, за которой не идет та фишка, которая должна бы идти (в целевой конфигурации), начисляется два очка, а в противном случае берется нуль очков.

За фишку, находящуюся в центре, начисляется одно очко.

2	1	6
4	■	8
7	5	3

1	2	3
8	■	4
7	6	5

Эвристический поиск

Используя функцию ***h(n) = P(n) + 3S(n)*** в оценочной функции

f(n) = g(n) + h(n)

(здесь ***g(n)*** — длина (стоимость) пути из начальной вершины ***s*** в вершину ***n)***

можно находить решения в достаточно сложных ситуациях игры в восемь.

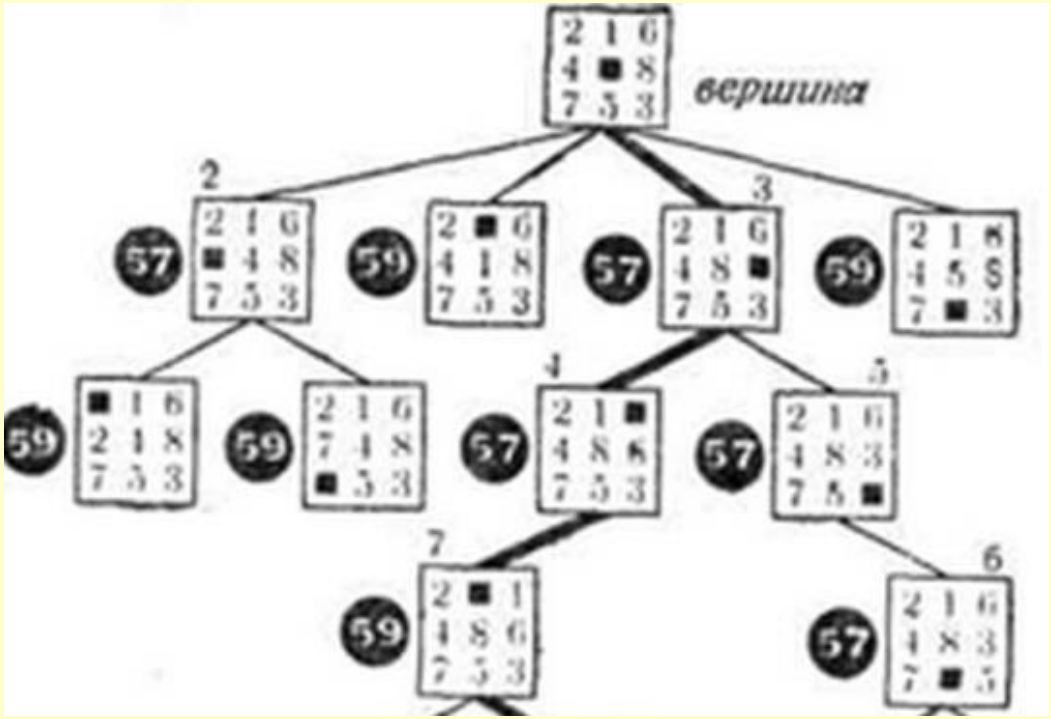
Эвристический поиск

Дерево, возникающее в результате применения алгоритма упорядоченного перебора с приведенной выше оценочной функцией к задаче преобразования левой конфигурации в правую.

2	1	6
4	■	8
7	5	3

➔

1	2	3
8	■	4
7	6	5



Эвристический поиск

Полное дерево перебора.

2	1	6
4	■	8
7	5	3

➔

1	2	3
8	■	4
7	6	5