Лекция 4.
Тема: Методы поиска решений в пространстве.

Методы поиска решений в пространстве состояний начнем рассматривать с простой задачи о миссионерах и людоедах. Три миссионера и три людоеда находятся на левом берегу реки и им нужно переправиться на правый берег, однако у них имеется только одна лодка, в которую могут сесть лишь 2 человека. Поэтому необходимо определить план, соблюдая который и курсируя несколько раз туда и обратно, можно переправить всех шестерых. Однако если на любом берегу реки число миссионеров будет меньше, чем число людоедов, то миссионеры будут съедены. Решения принимают миссионеры, людоеды их выполняют.

Основой метода являются следующие этапы.

1. Определяется конечное число состояний, одно из состояний принимается за начальное и одно или несколько состояний определяются как искомое (конечное, или терминальное). Обозначим состояние S тройкой S=(x,y,z), где x и y - число миссионеров и людоедов на левом берегу, z= {L,R} - положение лодки на левом (L) или правом (R) берегах. Итак, начальное состояние S_0=(3,3, L) и конечное (терминальное) состояние S_k=(0,0, R).
2. Заданы правила перехода между группами состояний. Введем понятие действия M:[u, v]w, где u - число миссионеров в лодке, v - число людоедов в лодке, w - направление движения лодки (R или L).
3. Для каждого состояния заданы определенные условия допустимости (оценки) состояний: x >= y ; 3-x >= 3-y ; u+v <= 2.
4. После этого из текущего (исходного) состояния строятся переходы в новые состояния, показанные на рис. 1. Два новых состояния следует сразу же вычеркнуть, так как они ведут к нарушению условий допустимости (миссионеры будут съедены).
5. При каждом переходе в новое состояние производится оценка на допустимость состояний и если при использовании правила перехода для текущего состояния получается недопустимое состояние, то производится возврат к тому предыдущему состоянию, из которого было достигнуто это текущее состояние. Эта процедура получила название *бэктрекинг* (bac tracing или ***BACKTRACK***).

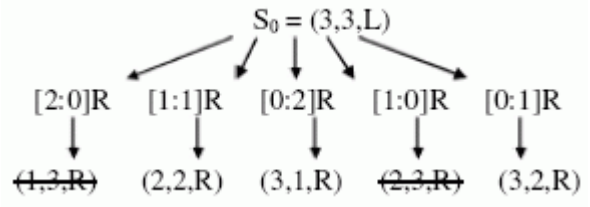

Рис. 1. Переходы из исходного состояния

Теперь мы можем проанализировать полностью *алгоритм* простейшего *поиска решений в проблемном пространстве*, описанный группами состояний и переходами между состояниями на рис. 2. Решение задачи выделено на рис. 2 жирными стрелками. Такой *метод поиска* S_0 => S_k называется прямым *методом поиска*. *Поиск* S_k => S_0 называют обратным поиском. *Поиск* в двух направлениях одновременно называют двунаправленным поиском.

Как уже упоминалось, фундаментальным понятием в *методах поиска* в ИС является идея рекурсии и процедура *BACKTRACK*. В качестве примера многоуровневого возвращения рассмотрим задачу размещения на доске 8 x 8 восьми ферзей так, чтобы они не смогли "съесть" друг друга.

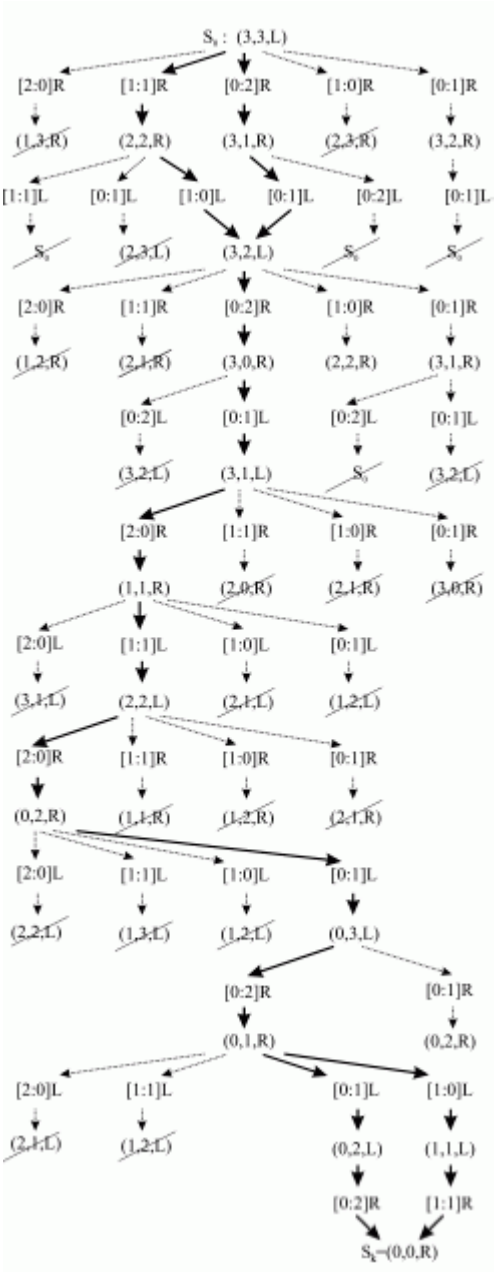

Рис. 2. Метод поиска в пространстве состояний

Допустим, мы находимся на шаге размещения ферзя в 6 ряду и видим, что это невозможно. Процедура *BACKTRACK* пытается переместить ферзя в 5 строке и в 6 строке опять неудача. Только возврат к 4 строке и нахождение в ней нового варианта размещения приведет к решению задачи. Читатель сам может завершить решение этой задачи на основе процедуры *BACKTRACK*.

x							
		x					
				x			
	x						
			x				

Алгоритмы эвристического поиска

В рассмотренных примерах поиска решений число состояний невелико, поэтому перебор всех возможных состояний не вызвал затруднений. Однако при значительном числе состояний время поиска возрастает экспоненциально, и в этом случае могут помочь *алгоритмы эвристического поиска*, которые обладают высокой вероятностью правильного выбора решения. Рассмотрим некоторые из этих алгоритмов.

Алгоритм наискорейшего спуска по дереву решений

Пример построения более узкого дерева рассмотрим на примере задачи о коммивояжере. Торговец должен побывать в каждом из 5 городов, обозначенных на карте (рис. 3).

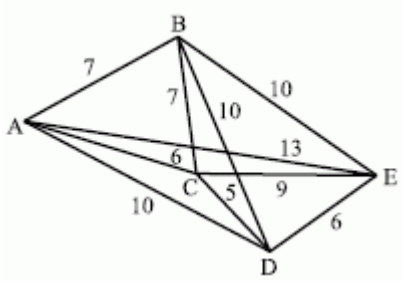

Рис. 3.

Задача состоит в том, чтобы, начиная с города A, найти минимальный путь, проходящий через все остальные города только один раз и приводящий обратно в A. Идея метода исключительно проста - из каждого города идем в ближайший, где мы еще не были. Решение задачи показано на рис. 4.

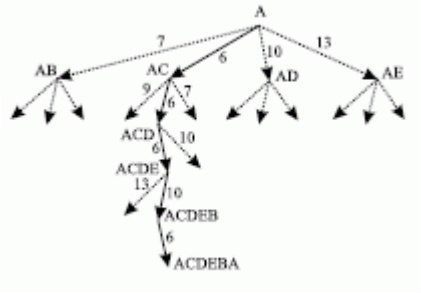

Рис. 4.

Такой алгоритм поиска решения получил название *алгоритма наискорейшего спуска* (в некоторых случаях - наискорейшего подъема).

Алгоритм оценочных (штрафных) функций

Умело подобранные оценочные функции (в некоторых источниках - штрафные функции) могут значительно сократить полный перебор и привести к решению достаточно быстро в сложных задачах. В нашей задаче о людоедах и миссионерах в качестве самой простой целевой функции при выборе очередного состояния можно взять число людоедов и миссионеров, находящихся "не на месте" по сравнению с их расположением в описании целевого состояния. Например, значение этой функции f=x+y для исходного состояния $f_0=6$, а значение для целевого состояния $f_1=0$.

Эвристические процедуры поиска на графе стремятся к тому, чтобы минимизировать некоторую комбинацию стоимости пути к цели и стоимости поиска. Для задачи о людоедах введем оценочную функцию:

f(n) = d(n) + w(n)

где d(n) - *глубина вершины* n на дереве поиска и w(n) - число находящихся не на нужном месте миссионеров и людоедов. *Эвристика* заключается в выборе минимального значения f(n). Определяющим в *эвристических процедурах* является выбор оценочной функции.

Рассмотрим вопрос о сравнительных характеристиках оценочных *целевых функций* на примере функций для игры в "8" ("пятнашки"). Игра в "8" заключается в нахождении минимального числа перестановок при переходе из исходного состояния в конечное (терминальное, целевое).

2	8	3
1	6	4
7	*	5

1	2	3
8	*	4
7	6	5

Рассмотрим две оценочные функции:

$h_1(n)$ & = Q(n)

$h_2(n)$ & = P(n) + 3S(n),

где Q(n) - число фишек не на месте; P(n) - сумма расстояний каждой фишки от места в ее целевой вершине; S(n) - учет последовательности нецентральных фишек (штраф +2 если за фишкой стоит не та, которая должна быть в правильной последовательности; штраф +1 за фишку в центре; штраф 0 в остальных случаях).

Сравнение этих оценочных функций приведено в таблица.

Таблица. Сравнение оценочных функций

Оценочная функция h	**Стоимость (длина) пути L**	**Число вершин, открытых при нахождении пути N**	**Трудоемкость вычислений, необходимых для подсчета h S**	**Примечания**
h_1 S_0 S_1	5 >18	13 100-8! (=40320)	8	*Поиск в ширину*
h_2 S_0 S_1	5 18	11 43	8*2+8+1+1	*Поиск в глубину*

На основе сравнения этих двух оценочных функций можно сделать выводы.

• Основу алгоритма поиска составляет выбор пути с минимальной оценочной функцией.

• *Поиск в ширину*, который дает функция h_1, гарантирует, что какой-либо путь к цели будет найден. При поиске в ширину вершины раскрываются в том же порядке, в котором они порождаются.

• *Поиск в глубину* управляется эвристической компонентой 3S(n) в функции h_2 и при удачном выборе оценочной функции позволяет найти решение по кратчайшему пути (по минимальному числу раскрываемых вершин). *Поиск в глубину* тем и характеризуется, что в нем первой раскрывается та вершина, которая была построена самой последней.

• Эффективность поиска возрастает, если при небольших глубинах он направляется в основном в глубь эвристической компонентой, а при возрастании глубины он больше похож на поиск вширь, чтобы гарантировать, что какой-либо путь к цели будет найден. Эффективность поиска можно определить как E=K/L*N*S, где K и S (трудоемкость) - зависят от оценочной функции, L - *длина пути*, N - число вершин, открытых при нахождении пути. Если договориться, что для оптимального пути E=1, то $K=L^0*N^0*S^0$.

Алгоритм минимакса

В 1945 году Оскар Моргенштерн и Джон фон Нейман предложили ***метод минимакса***, нашедший широкое применение в теории игр. Предположим, что противник использует оценочную функцию (ОФ), совпадающую с нашей ОФ. Выбор хода с нашей стороны определяется максимальным значением ОФ для текущей позиции. Противник стремится сделать ход, который минимизирует ОФ. Поэтому этот метод и получил название

минимакса. На рис. 5 приведен пример анализа дерева ходов с помощью *метода минимакса* (выбранный путь решения отмечен жирной линией).

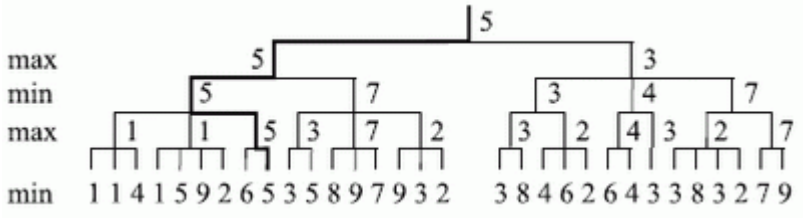

Рис. 5. Дерево ходов

Развивая *метод минимакса*, назначим вероятности для выполняемых действий в задаче о миссионерах и людоедах:

P([2 : 0]R) = 0; 8; P([1 : 1]R) = 0; 5;

P([0 : 2]R) = 0; 9;

P([1 : 0]R) = 0; 3; P([0 : 1]R) = 0; 3:

Интуитивно понятно, что посылать одного людоеда или одного миссионера менее эффективно, чем двух человек, особенно на начальных этапах. На каждом уровне мы будем выбирать состояние по критерию P_i. Даже такой простой подход позволит нам избежать части тупиковых состояний в процессе поиска и сократить время по сравнению с полным перебором. Кстати, этот подход достаточно распространен в экспертных продукционных системах.