4. Поиск в пространстве состояний. Формальная постановка задачи. Обобщенный алгоритм поиска. Вершины дерева поиска и состояния пространства состояний поиска. Операции над каймой.

Многие задачи, в частности игры и головоломки, могут быть представлены как задачи *поиска в пространстве состояний*.

Решить задачу – значит найти путь из исходного состояния в целевое.

Формально задача *поиска в пространстве состояний* в общем случае задается четверкой:

$$<I, \{O_i\}, GT, PC>,$$

где I – исходное состояние, т. е. состояние мира в начале задачи;

$\{O_i\}$ – множество действий (операторов), возможных в различных состояниях;

GT (*goal test*) – проверка достижения целевого состояния;

PC (*path cost*) – функция стоимости пути.

Действия (операторы) переводят состояния в другие состояния. Таким образом, можно ввести в рассмотрение *функцию последователей* S (successor), ставящую в соответствие каждому состоянию x множество состояний $S(x)$, достижимых из x за одно действие.

Исходное состоянии и множество действий (операторов) в совокупности определяют *пространство состояний задачи*, т.е. множество всех состояний, *достижимых* из исходного, путем конечной последовательности действий.

Проверка достижения целевого состояния позволяет для каждого достигнутого состояния определить является ли оно целевым. Существует два способа задания целевых состояний:

- явное перечисление множества целевых состояний;
- использование предикатов, описывающих целевые состояния.

Как правило, некоторое решение задачи является более предпочтительным, чем другие.

Функция *PC* (path cost) позволяет вычислить стоимость пути в заданных единицах. Как правило *PC* является аддитивной оценкой, т. е. стоимость пути вычисляется как сумма стоимостей элементов пути (операторов).

Для описания алгоритмов реализации поиска в пространстве состояний компоненты задачи должны быть представлены соответствующими данными:

datatype Problem component : INITIAL-STATE, Operations, Goal Test, Path-Cost-Funct.

Эффективность того или иного *метода поиска* определяется ответами на следующие вопросы:

1. Находит ли он решение в принципе (полнота);
2. Является ли найденное решение хорошим (т.е. имеющим низкую стоимость пути).
3. Какова стоимость реализации поиска, определяемая временем и памятью, требуемыми для нахождения решения (пути).

Полная стоимость поиска есть сумма стоимости пути и стоимости поиска пути. Решением задачи – является путь к целевому состоянию, имеющий конкретную стоимость. Чтобы найти этот путь нужно проделать поисковые действия, что требует времени и памяти.

Процесс поиска в пространстве состояний удобно рассматривать как *построение дерева поиска*, которое накладывается на пространство состояний.

Корнем этого дерева является вершина соответствующая начальному состоянию.

Листья дерева соответствуют состоянию, не имеющему потомков в дереве, либо из-за того, что они еще не были раскрыты, либо они сгенерировали пустое множество потомков.

На каждом шаге алгоритм поиска выбирает для раскрытия одну концевую вершину.

Общий алгоритм поиска можно представить следующим образом:

Function *General-Search* (Problem, Strategy) **returns** solution or failure

Инициализация дерева поиска исходным состоянием задачи I

Loop do - цикл

if (нет вершин - кандидатов для раскрытия)

then return failure

Выбрать концевую вершину (лист) для раскрытия в соответствии со стратегией.

if (вершина содержит целевое состояние)

then return solution (путь к этой вершине)

else

Раскрыть вершину и добавить новые вершины в дерево поиска.

end

Важно различать *пространство состояний* и *дерево поиска*. Для представления вершин в дереве поиска необходимо использовать структуры данных, имеющие следующие компоненты:

1. Состояние в пространстве состояний, которому сопоставлена вершина.

2. Родительская вершина – это вершина, непосредственным потомком которой является данная вершина.

3. Оператор, в результате применения которого была порождена данная вершина.

4. Глубина вершины – число вершин в пути от корня дерева к данной вершине.

5. Стоимость пути от корневой вершины к текущей.

Таким образом, необходимо различать *вершины* и *состояния* в пространстве поиска.

Состояние представляет собой элемент пространства состояний , т.е. некоторое состояние мира. Вершина – это структура данных, используемая для представления дерева поиска. Таким образом, вершина имеет глубину и родителей, а состояния не имеют.

Множество вершин, ожидающих раскрытия, принято называть *каймой* или границей (fringer).

Стратегия поиска представляет собой функцию, выбирающую из каймы (граничного множества вершин) очередную вершину для раскрытия.

С алгоритмической точки зрения граничное множество удобно представить очередью и определить над ней следующие операции:

1. make-Queue (Elements) – создание очереди с заданными элементами;

2. empty?(Queue) – возвращает true, если очередь пуста;

3. remove-front(Queue) – возвращает первый элемент очереди и удаляет его из очереди;

4. Queueing-Fn(Elements, Queue) – добавляет в очередь множество элементов. Именно эта функция определяет разные стратегии, реализующие разные алгоритмы поиска.

С использованием введенных обозначений можно записать алгоритм поиска:

```
function General-Search(Problem, Queuing-Fn) returns solution or failure
   nodes ← make-Queue(Make-Node(Init-State[Problem])
        //присваивание Make-Node;
        // порождение вершины, т.е. получили корень
    loop do
if  nodes = 0 then return failure
node ← Remove-Front (nodes)
        if Goal-Test[problem] (State(node))=true
 then return node
 else nodes ← Queueing-Fn(nodes, Expand(node, OPERATORS[Problem]))
end
```

Последний оператор к множеству уже существующих вершин добавляет результат раскрытия Expand. OPERATORS обеспечивают переход из одного состояния в другое. Результатом являются все возможные потомки в вершине Expand.