# 3. РЕШЕНИЕ ЗАДАЧ МЕТОДОМ РАЗБИЕНИЯ НА ПОДЗАДАЧИ

Идея метода состоит в следующем. Если путь решения задачи неясен (или неизвестен), то нужно постараться найти какую-то точку опоры, от которой (или до нее) путь известен. Это точка может быть в любом месте - в начале, в середине, в конце, но она должна быть. Этой точке, очевидно, будет соответствовать некоторое состояние ***Si*** на пути от *Sн* до *Sц*. Если эта точка *Si* определена, то к ней уже можно применить какое-либо действие и этим изменить состояние предметной области, т.е. приблизиться к возможному решению. Для нового состояния также ищется точка опоры на оставшемся пути, к ней снова применяется какое-то (или то же самое) действие и т.д., пока не будет решена вся задача.

Чтобы детально описать этот метод, необходимо вернуться к определению понятия задачи. Для этой цели воспользуемся понятием пространства состояний. В более общем, чем прежде, виде задачу (*Z*) можно представить следующим образом:

$$Z = \langle S, G, F \rangle, \qquad (3.1)$$

где *S* - множество начальных состояний, *G* - множество операторов, переводящих предметную область из одного состояния в другое, *F* -множество целевых состояний.

При таком обозначении промежуточные состояния *Si* удобнее обозначать через *fi*, поскольку они теперь представляют собой как бы промежуточные цели.

Конечной целью сведения задачи к подзадачам является получение таких элементарных задач, решение которых очевидны. *Элементарными* считаются задачи, которые могут быть решены за один шаг, т.е. за одно применение какого-либо оператора из множества *G*.

### *3.1. Представление задачи в виде И/ИЛИ графа*

Между полученными при разбиении подзадачами могут быть отношения согласованности (одновременности) их решения (отношение ***И***), или отношение альтернативности (отношение ***ИЛИ***). Рассмотрим это подробнее.

Допустим, что исходная задача *Z* может быть разбита на подзадачи *A*, *B*, *C*, *D*, *E*. Пусть при этом подзадачи *A* и *B*, а также *C* и *D* находятся в отношениях ***И***, т.е. должны решаться согласованно (одновременно), а множества (*A* ***И*** *B*), (*C* ***И*** *D*) и подзадача *E* находятся в отношении ***ИЛИ*****,** т.е. имеют альтернативный характер отношений. Тогда все сказанное легко отобразить на графе (рис.3.1)

.

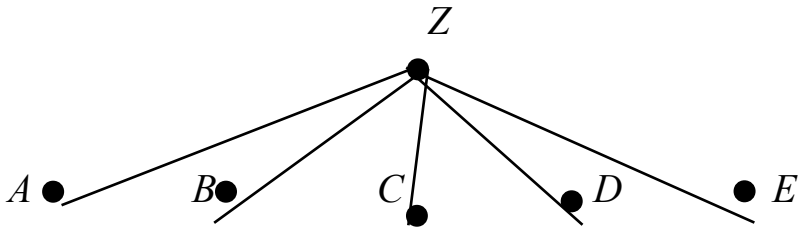


*Рис. 3.1. Граф отношений подзадач для задачи Z*

Отношения типа ***И*** будем отмечать двойной дугой, связывающей ребра графа. Для единообразия представления можно ввести дополнительные (фиктивные) вершины, которые группировали бы подзадачи типа ***И*** и ***ИЛИ*** под своей собственной *родительской* вершиной (рис. 3.2).

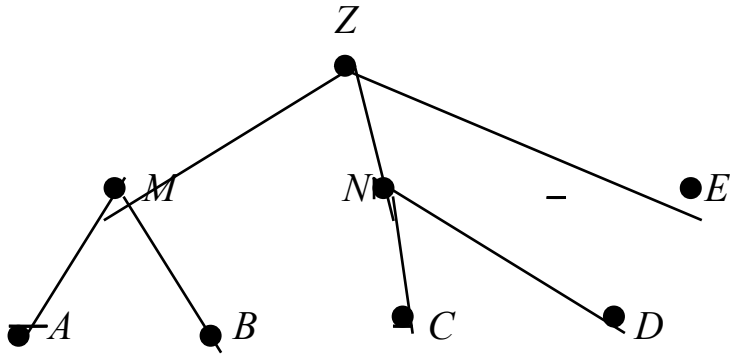


*Рис. 3.2. И/ИЛИ граф задачи Z*

Таким образом, исходная задача *Z* представляется подзадачами *M*, *N* и *E*, имеющими альтернативный *ИЛИ* - характер, а подзадачи *M* и *N*, в свою очередь, - подзадачами (*A*,*B*) и (*C*,*D*) с отношениями типа *И*. Альтернативные вершины *M*, *N*, *E* называются поэтому *ИЛИ* - вершинами, а вершины *A*, *B*, *C*, *D* - *И* - вершинами и помечаются двойной дугой.

Структура типа изображенной на рис. 3.2 называется *И*/*ИЛИ* графом. При этом, как видим, если вершина *И*/*ИЛИ* графа имеет непосредственно следующие за ней вершины, то либо все они являются *ИЛИ* - вершинами, либо вершинами *И*. Если у вершины

имеется только одна следующая за ней вершина, то ее можно считать как *И*, так и *ИЛИ* - вершиной. Если вершин типа *И* вообще нет, то получается обычный граф, тот самый, который получался в методе полного перебора при поиске в пространстве состояний.

Структура типа *И*/*ИЛИ* графа является удобной для представления дерева подзадач. При этом начальная вершина графа соответствует начальному состоянию *Sн* исходной задачи, а вершины, которые соответствуют описаниям элементарных подзадач, называются *заключительными*.

Основная цель поиска на *И*/*ИЛИ* графе - показать *разрешимость* вершины *Si*. Вершина является *разрешимой*, если выполняется одно из следующих условий:

1) вершина *Si* является заключительной;

2) следующие за *Si* вершины являются вершинами типа *ИЛИ* и при этом хотя бы одна из них разрешима;

3) следующие за *Si* вершины являются вершинами типа *И* и при этом каждая из них разрешима.

*Решающим графом* называется подграф, состоящий из разрешимых вершин с корнем в начальной вершине.

В случае если у вершины *И*/*ИЛИ* графа, не являющейся заключительной, нет следующих за ней вершин, такая вершина называется *неразрешимой*.

Порождение новых вершин (а эта операция еще называется операцией *редукции задачи*) выполняется путем применения обобщенного оператора *G* (т.е. каких-либо операторов из множества *G*) сведения задачи к подзадачам. Применение оператора *G* к описанию задачи порождает всю структуру *И*/*ИЛИ* графа (или иначе *графа редукции*).

### *3.2. Механизм сведения задачи к подзадачам*

Напомним, что наша цель - построить некоторый алгоритм, который, будучи примененным чисто механически, обязательно даст результат, т.е. требуемую структуру - *И*/*ИЛИ* граф с выделенным решающим подграфом. Такой алгоритм представляет собой один из механизмов планирования решения задачи.

Как следует из предыдущего, если исходная задача *Z* задается своим начальным описанием

$$Z = \langle\ S,G,F\ \rangle,$$

то свести ее к совокупности более простых задач в пространстве состояний можно, если нам удастся выделить основные промежуточные состояния *f1*, *f2*, *f3*...*fn*. Каждому из этих состояний в соответствие можно поставить свое описание в виде троек

〈 *S*, *G*, *f1* 〉, 〈 *f1*, *G*, *f2* 〉, 〈 *f2*, *G*, *f3* 〉,...〈 *fn*, *G*, *F* 〉.

Решение этих подзадач эквивалентно решению исходной задачи. На этой идее построен основной механизм сведения задачи к подзадачам.

1. Выделяем по крайней мере один оператор $gi \in G$, который обязательно будет участвовать в решающей цепочке операторов. Все операторы такого типа называются *ключевыми* (т.е. операторы, обязательно участвующие в решающей цепочке).

2. Для каждого из ключевых операторов (если их несколько) определяются промежуточные состояния, к которым они могут быть применены в условиях задачи *Z*. Для оператора *gi* это будет состояние *fi*. (Таких состояний, вообще говоря, может быть несколько, и тогда они образуют подмножество целевых состояний $Fgi \in F$). Теперь уже можно выделить подзадачу поиска пути от начала до состояния *fi* (или до *Fgi*). Другими словами, применение оператора *gi* привело к первой подзадаче с описанием

〈 *S*, *G*, *fi* 〉 (или 〈 *S*, *G*, *Fgi* 〉).

3. Как только такое описание найдено, может быть сформулирована 2-я подзадача, соответствующая элементарной. В самом деле, поскольку состояние *fi* соответствует оператору *gi*, то ничто не мешает нам применить его к *fi* и получить, таким образом, новое состояние *gi*(*fi*), определенно приближающее нас к конечной цели, правда, только на один шаг.

4. От вновь полученного состояния *gi*(*fi*) до конечной цели *F* еще может быть неблизкий путь. Определить его – третья подзадача.

Таким образом, применение выбранного оператора *gi* к задаче с описанием *Z* = 〈 *S*, *G*, *F* 〉 позволяет выделить сразу три подзадачи: 〈 *S*, *G*, *fi* 〉, 〈 *fi*, *gi*, *gi* (*fi*) 〉, 〈 *gi*(*fi*), *G*, *F* 〉,

одна из которых элементарная. Сказанное удобно проиллюстрировать на линейном графике (рис. 3.3). Все решение изображается отрезком, который разбивается точкой *fi* , соответствующей оператору *gi* на подзадачи

| | *G* | | *gi* | *G* | |
|---|---|---|---|---|---|

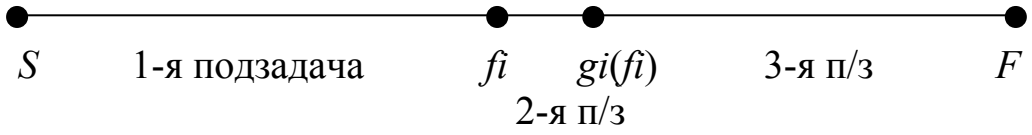


*Рис. 3.3. Разбиение задачи на подзадачи (п/з)*

Такому разбиению будет соответствовать следующий И/ИЛИ граф (рис.3.4).

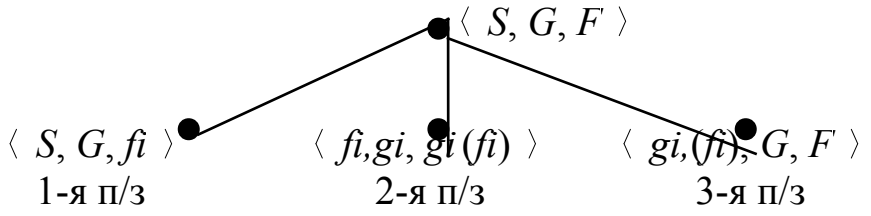


*Рис. 3.4. И/ИЛИ граф разбиения на подзадачи для состояния fi*

Элементарная подзадача типа 2 решается всегда для любой выбранной точки *f* пространства состояния, поэтому ее можно не указывать. Точка *fi* - одна из возможных промежуточных целей: $fi \in Fgi$. Выбрав ее, мы применяем к ней оператор *gi*. Обобщая сказанное, приходим к окончательному виду *И/ИЛИ* графа разбиения на подзадачи для одной точки (рис.3.5).

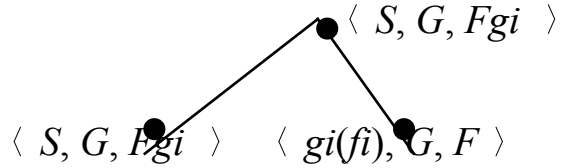


*Рис. 3.5. Обобщенный И/ИЛИ граф для одной точки (fi)*

5. Каждая из полученных при разбиении подзадач, если она, конечно, не элементарная, может быть снова разбита на подзадачи аналогичным методом , возможно, с помощью другого оператора.

*Примечание*. Возможны случаи, когда множество операторов *G* содержит всего один оператор *g*. Естественно, он является ключевым. Разбиение в этом случае начинается с удачного выбора промежуточного состояния *f* и далее по тому же алгоритму.

Итак, для разбиения задачи на подзадачи и построения соответствующего *И/ИЛИ* графа нужны ключевые операторы (обычно более одного). Один из способов нахождения операторов, могущих быть ключевыми, состоит в вычислении *различий* между состояниями по пути от $Sн \in S$ к $Sц \in F$. Каждому возможному различию ставится в соответствие оператор (или их множество),

который это различие может устранить. Цепочка операторов, последовательно устраняющих различия между $S_н$ и $S_ц$, называется решением задачи.

### *3.3. Пример решения задачи*

Проиллюстрируем метод разбиения задачи на подзадачи и поиска решений на *И/ИЛИ* графе на примере с обезьяной и бананами (см. п.1.5).

Допустим, что исходное описание задачи задается следующим образом:

$$S_н = (a,b,c,0,0) \in S,$$
$$S_ц = (c,c,c,1,1) \in F,$$
$$G=(g1, g2, g3, g4).$$

При этом отметим, что оператор $g_1$ устраняет различие между состояниями (*a,b,c,0,0*) и (*b,b,c,0,0*),

*g2* - между *(b,b,c,0,0)* и *(c,c,c,0,0)*,
*g3* - между *(c,c,c,0,0)* и *(c,c,c,1,0)*,
*g4* - между *(c,c,c,1,0)* и *(c,c,c,1,1)*.

Попытаемся далее выделить подзадачи, применяя, например, оператор *g3* (применение операторов может быть произвольным, поскольку речь идет о механической процедуре). Ему будут соответствовать подзадачи с описаниями:

1) *((a,b,c,0,0), G, (c,c,c,0,0))*, так как оператору *g3* соответствует состояние *Fg3=(c,c,c,0,0)*

2) *((c,c,c,0,0),g3, (c,c,c,1,0))*. Здесь *(c,c,c,1,0)=g3(c,c,c,0,0)*. То есть результат применения ключевого оператора *g3* к состоянию *(c,c,c,1,0)*.

3) Остается еще подзадача: *((c,c,c,1,0), G, (c,c,c,1,1))*.

Мы видим, что подзадача 1 не является заключительной (т.е. допускает дальнейшее разбиение), а подзадача 3 является элементарной, то есть заключительной, если применить оператор *g4*.

Подзадача 2 также является заключительной.

Разбиваем на подзадачи 1-ю подзадачу. Одно из различий между начальным и целевым состоянием может быть устранено применением, например, оператора *g2* (возможно и *g1*). Получим новые подзадачи с описаниями:

1.1. *((a,b,c,0,0), G, (b,b,c,0,0));*

1.2. *((b,b,c,0,0), g2, (c,c,c,0,0))*.

Здесь мы видим, что подзадача 1.2. является элементарной, а различие в подзадаче 1.1. может быть устранено, если применить оператор $g_1 \in G$, и тогда она является тоже элементарной.

Целиком граф И/ИЛИ приведен на рис. 3.6.

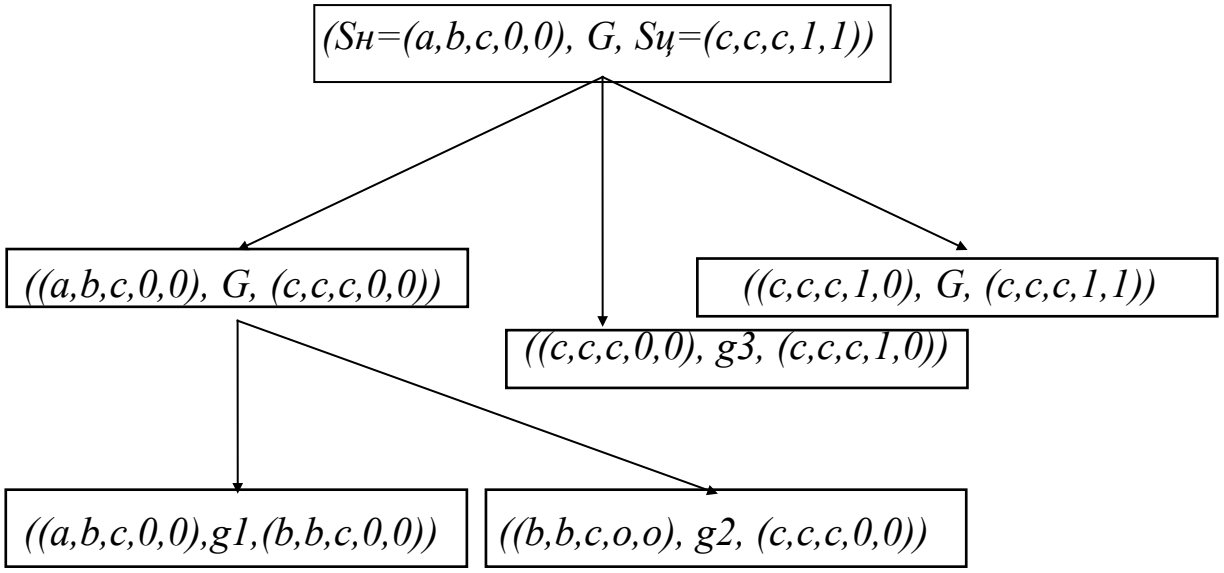


*Рис. 3.6. И/ИЛИ граф для задачи с обезьяной и бананами (ключевой оператор - g3)*

Точно так же можно было бы осуществить разбиение исходной задачи на подзадачи, применяя вначале любой другой оператор:*g4, g2,* или *g1*. На рис.3.7 показан граф для ключевого оператора *g4*.

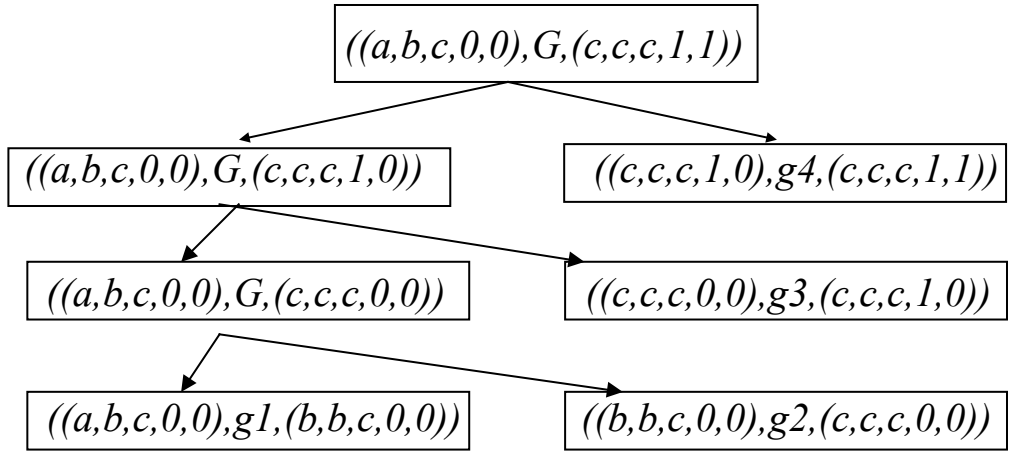


*Рис.3.7. И/ИЛИ граф для оператора g4*

### *3.4. Достоинства и недостатки методов поиска в пространстве состояний*

Приведем примеры задач, для решения которых могут применяться методы поиска в пространстве состояний.

1). *Комбинаторные задачи*. Классический пример – задача о коммивояжере (п. 2.4.). Состояния задаются списком городов, операторы соответствуют действию: направиться в город ***A***, направиться в город ***B***, … , направиться в город ***N***. Критерий достижения цели: любое описание, начинающееся и оканчивающееся городом ***A*** и перечисляющее все другие города, есть описание состояния, удовлетворяющего поставленной цели. Данная задача допускает поиск оптимального решения, поскольку дугам могут быть приписаны стоимости (длина пути) перехода из одного города в другой. Данная задача хорошо представима в графической форме (карте расстояний), что и делает ее удобной для применения методов поиска в пространстве состояний. Считается, что метод поиска в пространстве состояний применим при числе городов до 50. К данному классу задач хорошо сводятся практически любые однокритериальные задачи лабиринтного типа.

2). *Задачи синтаксического анализа*. При работе с языками (естественными или искусственными) задается грамматика построения правильных выражений (строк символов – слов, предложений, выражений) в данном языке и, следовательно, способ определения принадлежности произвольной строки символов к этому языку. Тогда множество состояний задачи синтаксического анализа может быть задано как множество строк (слов). Начальное и целевое состояния определяются какими-то фиксированными словами в алфавите этого языка. Операторы могут быть заданы в виде правил переписывания типа: μ ß æ → μ ø æ, где подстрока ß может быть заменена подстрокой ø. Эти правила могут выражать, например, синтаксис рассматриваемого языка. Критерий цели для этой задачи может быть задан строкой, число символов которой совпадает с числом символов целевого слова. Пример задачи такого типа: как слово «море» преобразовать в слово «рыба» путем замены на каждом шаге одной буквы. Очевидно, что при большом числе правил переписывания граф пространства состояний становится слишком большим.

3). *Задачи распределения*. Здесь известно: объем продукции каждого типа; величина поставок, которые должны быть осуществлены в заданные пункты. Необходимо найти такое распределение поставок, при котором, например, затраты на перевозки, были бы минимальны. Состояния описываются списком величин избыточной продукции, которая имеется в заданных пунктах. Операторы соответствуют передаче избытка продукции из одного пункта в другой. В качестве критерия может быть взято целевое состояние, при котором будут удовлетворены все заявки на поставку требуемой продукции. Типичная задача линейного программирования.

4). *Задачи управления* типа: требуется перевести объект управления с начальными значениями установленных параметров процесса в состояние, при котором эти параметры будут иметь заданные значения. Задача об обезьяне и бананах является типичной задачей этого класса.

Возникает вопрос, когда же следует применять описанные здесь методы поиска в пространстве состояний? Для каких классов задач эти методы применимы и могут дать эффективные решения?

**Во-первых**, это должны быть задачи, для которых такое представление возможно и является естественным. Это означает, что для выбранной задачи необходимо найти способ описания любых потенциально возможных состояний предметной области, определить множество операторов (ключевых операторов для метода разбиения задачи на подзадачи), с помощью которых эта предметная область может переходить из одного состояния в другое, определить критерий достижения цели.

**Во-вторых**, для этих задач не существуют или неизвестны методы, которые были бы более эффективны. Таким образом, мы допускаем, что одна и та же задача может решаться и другими методами, но по каким-то своим особенностям или соображениям эти методы представляются нам лучшими.

◆ Если найден удачный способ представления задачи, важно также, чтобы пространство состояний было не слишком большим. Существует множество примеров задач, кажущихся трудными, но таких, что при правильной их трактовке, соответствующие пространства состояний оказываются очень узкими. Подчас пространство состояний «сжимается в точку» после того, как

обнаруживается, что некоторые операторы могут быть выброшены за ненадобностью, а другие – объединены в более мощные операторы. И даже, если такие простые преобразования неосуществимы, может оказаться, что полная переформулировка задачи, изменяющая само понятие состояния, приведет к меньшему пространству. Таким образом, проблема поиска хорошего представления имеет важнейшее значение. Поиск его всегда опирается на специфику задачи, использует ее кардинальным образом. Это означает, что способ представления всегда носит уникальный характер, по сути, является особым искусством и, следовательно, вряд ли существуют какие-то единые правила поиска хороших представлений.

Во всех рассмотренных нами классах задач, для решения которых обычно применяются методы поиска в пространстве состояний, число свойств объектов предметной области и отношений между ними, служащих затем основой для распознавания ситуаций, всегда небольшое. Например, в задаче об обезьяне и бананах – это координаты (Обезьяны, Ящика, Бананов) и два отношения – «*Находиться НА*», «*Находиться У*». В задаче коммивояжера использовались: свойство «*Иметь имя*», а в качестве отношения – расстояние между каждыми двумя городами. В задаче распределения это: имена пунктов нахождения продукции и поставок, объемы имеющихся и требуемых поставок. Очевидно, что при увеличении числа свойств, отношений и их значений, размерность пространства состояний будет быстро возрастать таким образом, что применение переборных методов станет практически нереализуемым.

◆ В этой связи в искусственном интеллекте развивались и другие, более универсальные, модели представления знаний (МПЗ) о задаче. Общие требования к МПЗ можно сформулировать следующим образом:

1. Необходимы способы представления знаний о задаче, *безразличные к содержанию* (смыслу) самих знаний. Это позволит применять эти способы для представления знаний в любых предметных областях для решения задач.

2. Эти способы должны иметь *механизм логического анализа*, позволяющий моделировать человеческую логику выработки решений на выбранных моделях представления знаний.

3. Эти способы должны решать задачи, которые известными формальными моделями и методами не решаются в связи с их нестрогостью и нечеткостью.

Первое требование означает, что проблема смысла (содержания) знаний заменяется проблемой синтаксического их представления. Второе требование предполагает возможность разделения самих знаний и механизмов их логического анализа. Третье требование предполагает возможность формализации опытного (экспертного) знания, накапливаемого специалистами различных предметных областей и имеющих качественный (а не количественный) характер. Именно с таким знанием не могут работать все известные формальные математические модели. Мы, таким образом, приходим к необходимости разработки и применения *неформальных моделей знаний*.

***Вопросы для самопроверки и упражнения***

1. В каких случаях рекомендуется применять метод разбиения на подзадачи?
2. Дайте определение И-вершине, ИЛИ-вершине.
3. Какова структура И/ИЛИ графа?
4. Какая вершина называется заключительной?
5. Каково условие раскрытия И-вершины?
6. Каково условие раскрытия ИЛИ-вершины?
7. Какая вершина называется разрешимой?
8. Какой оператор является ключевым?
9. Объясните идею разбиения на подзадачи на линейном графике.
10. Постройте И/ИЛИ граф для задачи с обезьяной, начиная с оператора g2.
11. Задача о выборе маршрута.

Известно, что применение метода сведения задачи к подзадачам особенно эффективно, когда подзадачи взаимно

независимы. Покажем это на примере отыскания маршрута между городами *A* и *T* (см. рис. 3.8).

На рисунке задана маршрутная сеть между населенными пунктами, лежащими на двух берегах реки. *K* и *L* – это мосты, связывающие города одного берега с другим. На дугах между пунктами указаны расстояния (стоимость). При построении дерева подзадач следует учитывать, что добраться до из пункта *A* в пункт *T* можно двумя альтернативными путями: через *K* и через *L*. Постройте граф И/ИЛИ для данной задачи, определите кратчайшие пути.

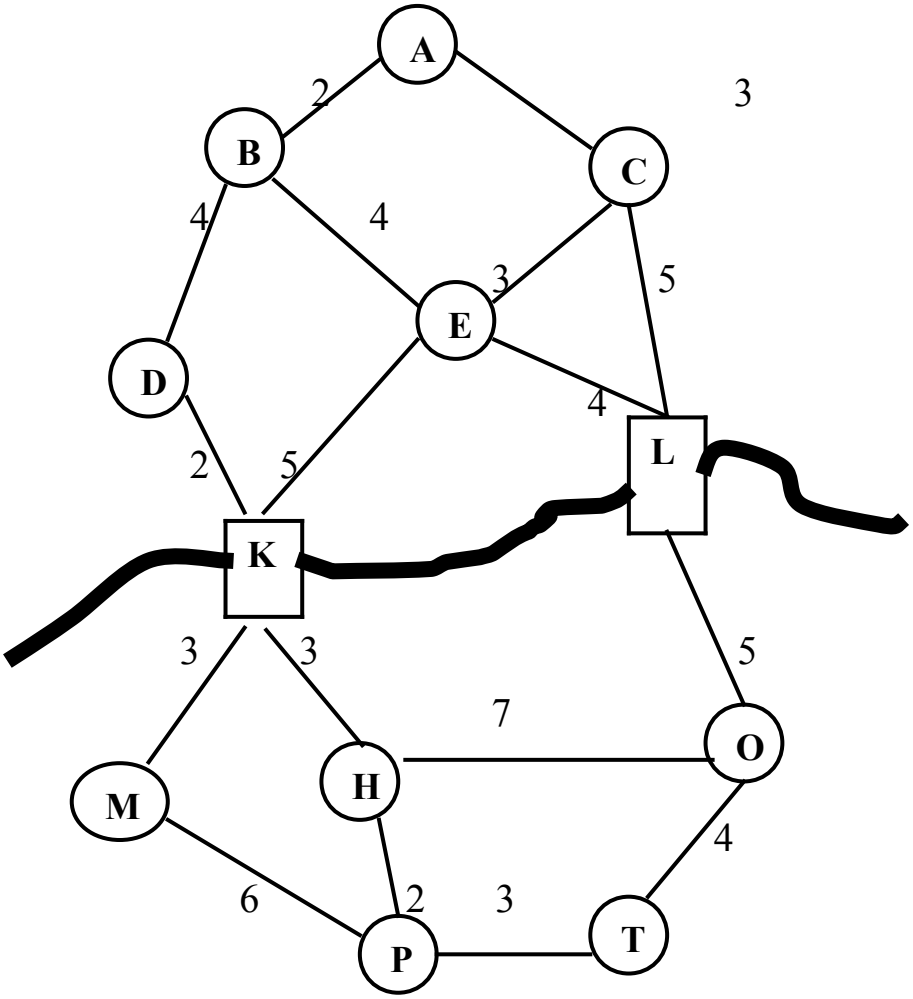


*Рис. 3.8. Задача о выборе маршрута*

12. Вернемся к задаче о квасе (гл. 2). Требуется разделить квас, находящийся в полном бочонке объемом 8 литров, пополам. Для этого имеются два пустых бочонка 5 и 3 л. Попробуйте решить эту задачу на этот раз методом сведения к подзадаче.

13. Постройте граф И/ИЛИ для задачи о коммивояжере (п. 2.4), начиная разбиение, к примеру, с пункта ***E.***