5. МУРАВЬИНЫЕ АЛГОРИТМЫ

5.1. Алгоритмы «природных вычислений»

В последние годы интенсивно разрабатывается научное направление Natural Computing — «Природные вычисления», объединяющее математические методы, в которых заложены принципы природных механизмов принятия решений. Эти механизмы обеспечивают эффективную адаптацию флоры и фауны к окружающей среде на протяжении миллионов лет. К этому направлению относятся методы роевого интеллекта (муравьиные, пчелиные) и методы эволюционной оптимизации (генетические).

Имитация самоорганизации муравьиной колонии составляет основу муравьиных алгоритмов — нового перспективного метода оптимизации. Колония муравьев может рассматриваться как мультиагентная среда, в которой каждый агент (муравей) функционирует автономно по очень простым правилам. В противовес почти примитивному поведению агентов, поведение всей системы получается на удивление разумным.

Коллективная система способна решать сложные динамические задачи по выполнению совместной работы, которая не могла бы выполняться каждым элементом системы в отдельности в разнообразных средах без внешнего управления, контроля или координации. В таких случаях говорят о роевом интеллекте (swarm intelligence), как о способах кооперативного поведения, то есть стратегии выживания. Поведение муравьев при транспортировании пищи, преодолении препятствий, строительстве муравейника и других действиях зачастую приближается к теоретически оптимальному.

Основу «социального» поведения муравьев составляет самоорганизация

— множество динамических механизмов, обеспечивающих достижение системой глобальной цели в результате низкоуровневого взаимодействия ее элементов. Принципиальной особенностью такого взаимодействия является использование элементами системы только локальной информации.

Самоорганизация является результатом взаимодействия следующих компонентов:

- случайность;
- многократность;
- положительная и отрицательная обратные связи;

Муравьи используют два способа передачи информации:

- прямой (обмен пищей и т.п.);
- непрямой, т.н. стигмержи (stigmergy).

Стигмержи — это разнесенный во времени тип взаимодействия, когда один агент взаимодействия изменяет некоторую часть окружающей среды, а остальные используют эту информацию позже, когда находятся в ее окрестности. Биологически непрямой обмен информациеей осуществляется через феромон (pheromone) — специальный секрет, откладываемый как след при перемещении муравья. Феромон — достаточно стойкое вещество, он может восприниматься муравьями несколько суток. Чем выше концентрация феромона на тропе, тем больше муравьев будет по ней двигаться. Со временем феромон испаряется, что позволяет муравьям адаптировать свое поведение под изменения внешней среды. Распределение феромона по пространству передвижения муравьев является своего рода динамически изменяемой глобальной памятью муравейника. Любой муравей в фиксированный момент времени может воспринимать и изменять лишь одну локальную ячейку этой глобальной памяти.

Таким образом, в общем случае рассматриваются слепые муравьи, не способные чувствовать близость пищи, но ориентирующиеся по запаху феромона.

5.2. Концепция муравьиных алгоритмов

Муравьиные алгоритмы представляют собой вероятностную жадную эвристику, где вероятности устанавливаются, исходя из информации о качестве решения, полученной из предыдущих решений.

Идея муравьиного алгоритма заключается в моделировании способности муравьев быстро находить кратчайший путь от муравейника к источнику пищи и адаптироваться к изменяющимся условиям, находя новый кратчайший путь. Муравей метит путь феромоном, и эта информация используется другими муравьями для выбора пути. Это элементарное правило поведения и определяет способность муравьёв находить новый путь, если старый оказывается недоступным.

В качестве примера рассмотрим случай, показанный на рис. 5.1, когда на оптимальном пути возникает преграда. В этом случае необходимо определение нового оптимального пути. Дойдя до преграды, муравьи с равной вероятностью будут обходить её справа и слева. Однако, те муравьи, которые случайно выберут кратчайший путь, будут быстрее его проходить, и за несколько циклов передвижений он будет более обогащён феромоном. Поскольку движение муравьёв определяется концентрацией феромона, то следующие муравьи будут предпочитать именно этот путь, продолжая обогащать его феромоном.

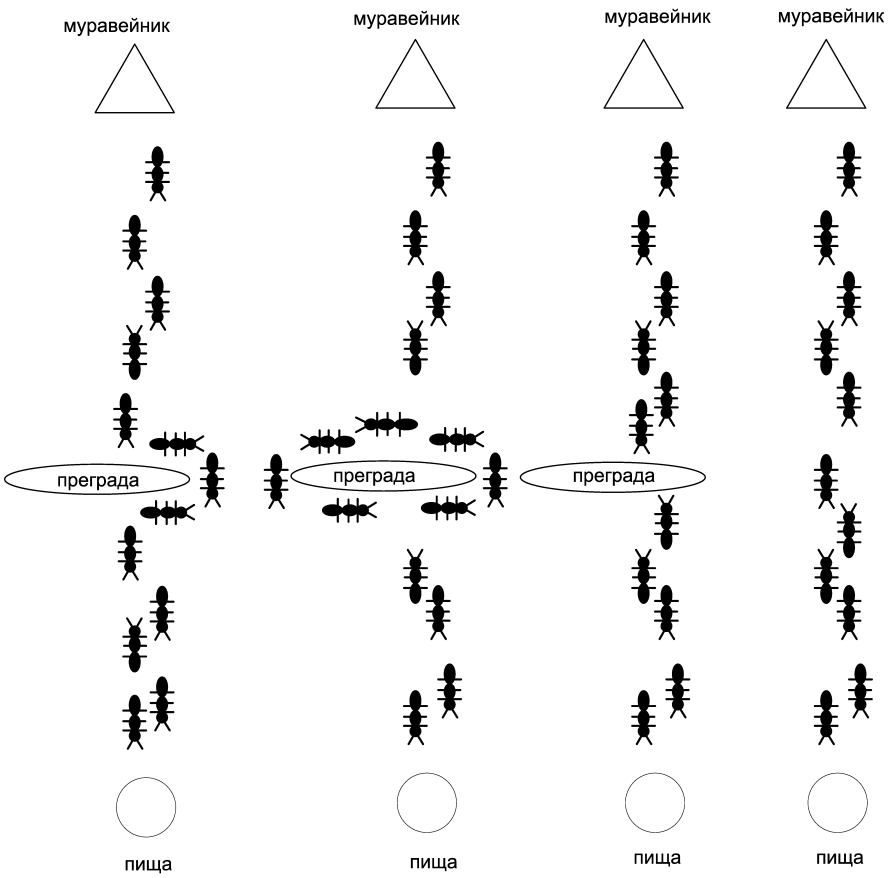

Рис. 5.1. Пример нахождения муравьями нового пути

Положительная обратная связь быстро приведёт к тому, что кратчайший путь станет единственным маршрутом движения большинства муравьёв. Испарения феромона – отрицательная обратная связь – гарантирует, что найденные неоптимальные решения будут терять свою привлекательность и муравьи будут искать другие пути. Допустим, что мы моделируем процесс такого поведения на некотором графе. Рёбра графа представляют собой возможные пути перемещения муравьёв в течение определённого времени. Тогда наиболее обогащённый феромоном путь по рёбрам графа и будет являться решением задачи, полученным с помощью муравьиного алгоритма.

5.3. Обобщенный муравьиный алгоритм

Любой муравьиный алгоритм, независимо от модификаций, представим в следующем виде

Циклическое выполнение этапов поиска (пока условия выхода не выполнены):

- Формирование колонии муравьёв;
- Поиск целевых вершин;
- Обновление феромона;

Рассмотрим последовательно каждый этап алгоритма:

0. Задание начальных условий поиска.

Задаём начальный одинаковый уровень феромона на ребрах графа. Он инициализируется небольшим положительным числом τ_0 для того, чтобы на начальном шаге вероятности перехода в следующую вершину не были нулевыми.

1. Формирование колонии муравьёв.

Создаем колонию из N муравьев и помещаем их в начальную вершину графа – b_0.

2. Поиск целевых вершин.

Вероятность перехода из вершины *bi* в вершину *bj* определяется по следующей формуле

$$P_{ij}(t) = \tau_{ij}(t) / \tau_{\Sigma}(t), \qquad \tau_{\Sigma}(t) = \sum_{l \in J_i} \tau_{il}(t) \tag{5.1}$$

где τ_{ij} - уровень феромона на ребре *ij* на итерации *t*, J_i- множество потомков вершины *i*.

Выражение (5.1) определяет лишь вероятности выбора того или иного ребра. Выбор для любого муравья каждый раз осуществляется по принципу «колеса рулетки». Если муравей находится в вершине *bi*, то выбор следующей вершины происходит случайным образом. Например, каждому ребру графа (рис. 5.2) назначается сектор на рулетке единичной площади. Площадь сектора пропорциональна вероятности (5.1) перехода по этому ребру. Для каждого муравья, находящегося в вершине *bi* формируется случайное число из диапазона (0,1), выпадающее на один из секторов. Муравей направляется по тому ребру, которое соответствует выпавшему сектору.

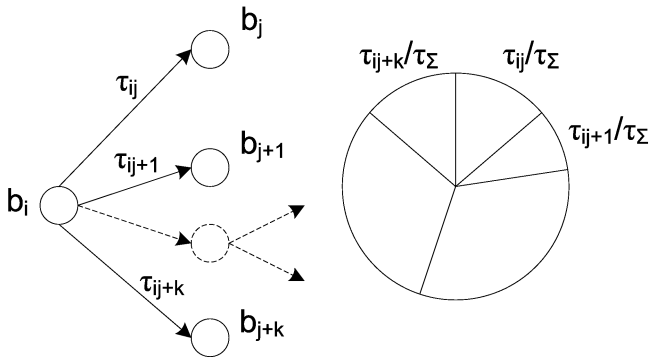

Рис. 5.2 Выбор вершины в муравьином алгоритме.

3. Обновление феромона.

Только после достижения целевой вершины каждый муравей *k*, прошедший по ребру *ij* откладывает на нем некоторое количество феромона:

$$\Delta\tau_{ij,k}(t) = Q / L_k(t)$$

где $L_k(t)$ — длина маршрута, пройденного муравьем k на итерации t; Q=const – запас феромона одного муравья на один маршрут. Если муравей не достигает целевой вершины, а заканчивает маршрут в «тупиковой» вершине, то он не откладывает феромон на пройденных ребрах.

Для исследования всего пространства решений необходимо обеспечить испарение феромона — уменьшение во времени количества отложенного на предыдущих итерациях феромона. Обозначим коэффициент испарения феромона через $p \in [0,1]$. Тогда правило обновления феромона на следующей итерации поиска примет вид:

$$\tau_{ij}(t+1) = (1-p)\tau_{ij}(t) + \Delta\tau_{ij}(t),$$

где $\Delta\tau_{ij} = \sum_{k \in K_{ij}} \Delta\tau_{ij,k}(t)$, K_{ij} – множество муравьев, прошедших по ребру *ij* на итерации *t*.

После завершения цикла поиска – t вся колония из N муравьев помещается вновь в начальную вершину b_0, (шаг 1), откуда начинается следующий цикл поиска $t+1$. Общее количество муравьев в колонии остается постоянным на протяжении выполнения алгоритма.

Многочисленная колония муравьев приводит к быстрому усилению

субоптимальных маршрутов, а когда муравьев мало, возникает опасность потери кооперативности поведения из-за их ограниченного взаимодействия и быстрого испарения феромона.

Поскольку в основе муравьиного алгоритма лежит моделирование передвижения муравьёв по некоторым путям, то такой подход может стать эффективным способом поиска рациональных решений для задач оптимизации, допускающих графовую интерпретацию. Эффективность муравьиных алгоритмов растёт с ростом размерности решаемых задач оптимизации. На основе применения муравьиных алгоритмов получены хорошие результаты для таких сложных оптимизационных задач, как задача коммивояжёра, транспортная задача, задача календарного планирования, задача раскраски графа, квадратичная задача о назначениях, задача оптимизации сетевых трафиков и ряд других.

6. ПОИСК НА ГРАФЕ

6.1. Графы и деревья

Особенность стратегии локального поиска состоит в том, что состояния среды, полученные на предыдущих шагах, забываются. Поэтому часто приходится заново порождать те состояния, которые встречались раньше. С целью устранения этого недостатка в стратегиях поиска применяют графовые структуры, сохраняющие информацию о всех получаемых состояниях. Стратегия поиска в данном случае создает граф, вершинами которого являются порожденные ею состояния b_i (i=0,1,2,...,u), а связи между вершинами определяются выполненными действиями агента. Суть решения задачи при этом сводится к поиску пути на графе от начальной вершины b_0 к целевой вершине bu.

Как известно, граф представляет собой непустое множество вершин вместе с множеством соединяющих их ребер (дуг). Вершины с ненаправленными связями образуют неориентированный граф, а с

направленными - ориентированный или орграф (рис.6.1). В этом графе b_1 - родительская вершина для вершин b_2, b_3; b_2 - вершина-преемник (потомок), или дочерняя вершина для b_1 и вершина-родитель для b_3; b_3 - вершина-преемник (потомок), или дочерняя вершина для b_1 и b_2. У ориентированных графов отношения родства между вершинами могут быть неоднозначными. Например, вершина b_3 одновременно дочерняя и «внучатая» по отношению к b_1.

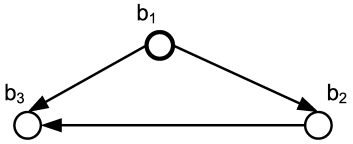

Рис. 6.1 Граф

Дерево - частный случай графа, в котором каждая вершина имеет не более одного родителя (рис. 6.2). Вершина, не имеющая родителя, называется корневой, а вершины, не имеющая потомков – концевыми (листьями дерева):

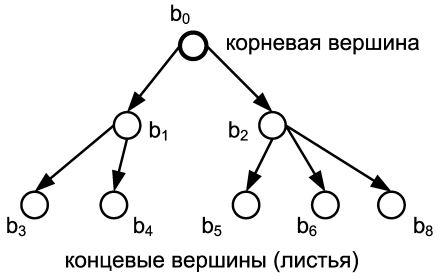

Рис. 6.2 Дерево

Глубина корневой вершины, как правило, принимается равной нулю. Глубина любой другой вершины равна глубине родительской вершины плюс единица. Путь на дереве глубиной m от b_0 до b_k есть последовательность таких вершин, для которых каждая последующая вершина b_{j+1} является преемником предыдущей вершины b_j.

Иногда дугам или ребрам приписывают положительные числа, соответствующие стоимостям соответствующих действий (цена связи между вершинами). Стоимость пути между вершинами b_0 и b_k равна сумме стоимостей всех дуг, лежащих на пути между ними. При наличии нескольких

путей между b_0 и b_k часто приходится искать путь минимальной стоимости.

Если концевая вершина b_k является элементом некоторого множества $\{b_k\}$ концевых вершин, удовлетворяющих целевым условиям, то множество $\{b_k\}$ называется целевым, а любой его элемент b_k называется целевой вершиной. Граф может быть определен явно и неявно. При явном определении граф задается таблицей вершин, дуг и их стоимостей. Однако при больших размерностях графов возникают трудности их явного задания. Поэтому часто используют неявное определение графов. При неявном определении задается начальная вершина b_0 и перечень возможных действий агента (операторов построения преемников). Процедура поиска раскрывает вершины с помощью операторов и порождает явное представление графа (например, граф игры в «восемь», в шашки, шахматы и т.п.). Иными словами, стратегия поиска на графе одновременно с поиском пути между вершинами b_0 и b_k порождает явно заданный подграф поиска на основе неявно определенного графа решения задачи в пространстве состояний среды.

6.2. Общая процедура поиска на графе

Рассмотрим общую процедуру поиска по графе. Процесс явного порождения части неявно заданного графа можно неформально определить следующим образом.

Процедура ГРАФ

1. Организовать список, называемый ***ОТКРЫТ (СпО)***, и занести в него начальное состояние среды b_0 (b_0 -> ***СпО***).
2. Создать список, называемый ***ЗАКРЫТ (Сп3)***, и обнулить его ***(Сп3=О)***.
3. Если список ***ОТКРЫТ*** пуст, то выдать сигнал **НЕУДАЧА** и перейти на конец (метка 10), иначе следующий шаг.
4. Выбрать первую вершину в списке ***ОТКРЫТ***, удалить ее из этого списка и поместить в список ***ЗАКРЫТ***, обозначив через b_k .
5. Если $b_k = b_Ц$, то зафиксировать **УСПЕХ**, выдать путь, ведущий от b_0 к $b_Ц$ и перейти на конец (метка 10), иначе следующий шаг.

6. Раскрыть вершину b_k , порождая множество $\{b_M\}_k$ преемников, не являющихся предками b_k и поместить их в список ***ОТКРЫТ***, если они ранее не были в него записаны.
7. Ввести указатели от введенных в ***CnO*** вершин b_M из множества $\{b_M\}_k$ к вершине - предку b_k, т.е. $b_M(b_k)$. При необходимости произвести переориентацию указателей уже имеющихся в списке вершин.
8. Переупорядочить список ***ОТКРЫТ*** в соответствии с эвристической значимостью вершин.
9. Перейти на метку 3.
10. **КОНЕЦ**

Процедура имеет достаточно общий характер и заключает в себе большое разнообразие отдельных алгоритмов поиска на графе. Процедура порождает в явной форме граф *G*, называемый графом поиска и подмножество *T* графа *G*, называемое деревом поиска. Каждая вершина из *G* содержится также и в *T*. Дерево поиска определено указателями, которые устанавливаются на шаге 7. Каждая вершина (за исключением начальной b_0) имеет указатель, направленный только к одному из родителей в *G*, который определяет ее единственного родителя в *T*.

Каждый возможный путь к какой-либо вершине, найденный этим алгоритмом хранится в явном виде в *G*. Один выделенный путь к любой вершине определен на дереве *T*.

На шаге 3 процедуры ГРАФ вершины в списке ОТКРЫТ являются теми (концевыми) вершинами дерева поиска, которые еще не выбирались для раскрытия. Вершины в списке ЗАКРЫТ являются либо неконцевыми вершинами дерева поиска, либо концевыми, уже выбранными для раскрытия и не породившими преемников в графе поиска.

На шаге 8 процедура ГРАФ упорядочивает вершины в списке ОТКРЫТ так, чтобы «лучшая» из них была выбрана для раскрытия на шаге 4. Это упорядочение может основываться на эвристических функциях оценки

состояний или на произвольных критериях. Всякий раз, когда вершиной, выбранной для раскрытия, оказывается целевая вершина, процесс успешно завершается. Решающий путь от исходной вершины к целевой можно затем восстановить, прослеживая (в обратном порядке) указатели от целевой вершины к b_0. Процесс заканчивается неудачей, когда на дереве поиска не остается концевых вершин, еще не выбранных для раскрытия. Если процесс завершается неудачей, то целевая вершина не может быть достижима из начальной.

Шаг 7 процедуры ГРАФ нуждается в дополнительном пояснении. Если бы неявно заданный граф, на котором ведется поиск, был деревом, мы могли бы быть уверены, что никакой из преемников, порожденных на шаге 6, не порождался бы ранее. Каждая вершина дерева является преемником только одной вершины и поэтому порождается один раз – когда раскрывается его единственная вершина - родитель. Граф G в этом случае совпадает с деревом T и нет необходимости менять родителей вершин в подмножестве Т.

Если же рассматриваемый неявно заданный граф не является деревом, то возможно, что некоторые вершины, порожденные на шаге 6, уже порождались, т.е. они уже могут быть в списке ОТКРЫТ или ЗАКРЫТ. Когда процесс поиска порождает вершину, уже возникавшую ранее, он находит к ней другой путь (возможно лучший). Желательно, чтобы дерево поиска сохраняло тот из найденных путей от b_0 к любой другой его вершине b_k, стоимость которого минимальна. Если вновь найденный путь короче (дешевле) прежнего, дерево поиска преобразуется – родительские функции утверждаются за последним родителем вновь порожденной вершины (рис.6.3).

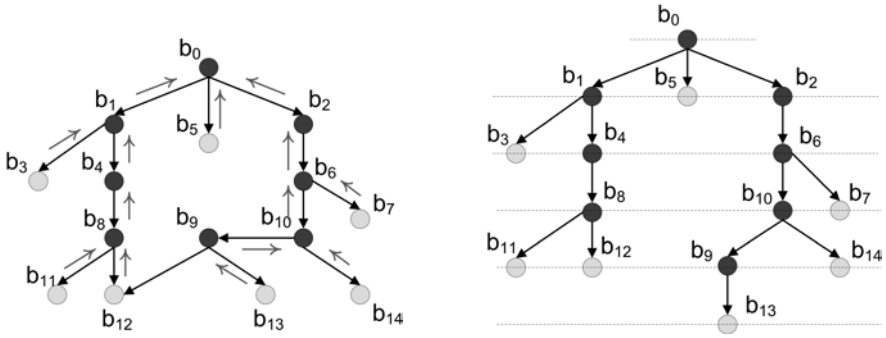

Рис. 6.3 Граф *G* и дерево *T* поиска до раскрытия вершины b_5

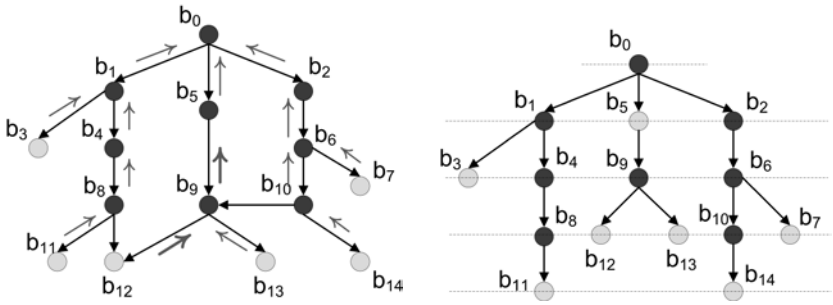

Рис. 6.4 Граф *G* и дерево *T* поиска после раскрытия вершины b_5

Вершины, закрашенные темной заливкой на рис 6.3 находятся в списке ЗАКРЫТ, а вершины закрашенные светлой заливкой – в списке ОТКРЫТ. До раскрытия вершины b_5 вершина b_{12} была преемником вершины b_8, потому что из двух путей от b_{12} до начальной вершины b_0 более коротким (4 ребра) является путь (b_{12}-b_8-b_4-b_1-b_0). Второй путь (b_{12}-b_9-b_{10}-b_6-b_2-b_0) более длинный (5 ребер). После раскрытия вершины b_5 вновь порождается потомок b_9, который уже находится в списке ЗАКРЫТ, но при этом алгоритм находит новый более короткий путь (3 ребра) между b_{12} и b_0 – (b_{12}-b_9-b_5-b_0). Происходит переориентация указателей на родителей у вершин b_{12} и b_9. Рядом показаны деревья поиска *T*, соответствующие на каждом этапе графу поиска *G*. На графе *G* отображены все пути между любой вершиной графа и начальной вершиной, а на дереве *T* только кратчайшие пути.

При отсутствии эвристической информации, можно произвольно устанавливать очередность раскрываемых вершин. Различают неинформированные стратегии поиска в глубину и стратегию полного перебора (поиск в ширину). Первый тип неинформированного поиска

располагает вершины списка ОТКРЫТ в порядке убывания их глубины в дереве поиска. Наиболее глубокие вершины помещаются в списке на первое место. Вершины, расположенные на одинаковой глубине, упорядочиваются произвольно. Процесс поиска, являющийся результатом такого упорядочения, называется поиском в глубину, поскольку для раскрытия всегда выбирается наиболее глубоко расположенная вершина дерева поиска. Для предупреждения неограниченного ухода процесса поиска по бесперспективному пути вводится ограничение на глубину. Вершины, глубина которых на данном дереве поиска превышает эту границу, вообще не порождаются.

Второй тип неинформированной процедуры поиска располагает вершины списка ОТКРЫТ в порядке возрастания их глубины в дереве поиска. Процесс поиска, являющийся результатом такого упорядочения, называется поиском в ширину, поскольку раскрытие вершины в дереве поиска происходит вдоль "уровня" на одинаковой глубине.

6.3. Эвристические процедуры поиска на графе

Для многих задач имеется возможность использовать некоторую информацию, относящуюся к рассматриваемой задаче, чтобы содействовать сокращению поиска. Информацию такого рода обычно называют эвристической. Некоторые эвристики существенно уменьшают затраты на поиск, но не гарантируют нахождения пути минимальной стоимости. В большинстве практических задач ИИ стремятся к тому, чтобы минимизировать некоторую комбинацию стоимости пути к цели и стоимости поиска, необходимого для нахождения этого пути. Более того, наиболее интересны такие методы поиска, которые минимизируют эту комбинацию, усредненную по всем ожидаемым задачам. Если усредненная стоимость комбинации для первого метода поиска меньше, чем для второго, то говорят, что первый метод поиска имеет большую эвристическую силу, чем второй.

Усредненные стоимости комбинаций в действительности никогда не

вычисляются. Во-первых, потому, что трудно принять решение, каким образом комбинировать стоимость пути и стоимость поиска. Во-вторых, потому, что трудно определить распределение вероятностей на множестве задач, с которыми придется столкнуться. Поэтому вопрос о том, обладает ли один метод поиска большей эвристической силой по сравнению с другим, обычно решается на основе интуиции, накопленной в процессе работы с этими методами.

Частным случаем эвристического поиска является поиск с использованием оценочных функций. Эвристическую информацию можно использовать для упорядочения вершин в списке ОТКРЫТ на шаге 8 процедуры ГРАФ таким образом, чтобы процесс поиска распространялся по тем участкам графа, которые представляются наиболее перспективными. Чтобы применить такую процедуру упорядочения, нужен метод вычисления "перспективности" вершины. Один из таких методов предусматривает использование функции, называемой оценочной и принимающей на вершинах действительные значения. В основе оценочных функций лежат самые разнообразные идеи:

- определить вероятность того, что некоторая вершина принадлежит наилучшему пути;
- предложить меры расстояния или различия между произвольной вершиной и целевым множеством;
- в настольных играх или головоломках для рассматриваемой конфигурации подсчитать число очков на основании тех особенностей, которые связаны с перспективностью этой конфигурации.

Используем функцию некоторую $f(b_i)$ для упорядочения вершин в списке ОТКРЫТ на шаге 8 процедуры ГРАФ. Условимся, что вершины в списке ОТКРЫТ расположены в порядке возрастания соответствующих значений функции $f(b_i)$. При совпадении значений $f(b_i)$ упорядочение

осуществляется произвольно, но целевым вершинам $b_ц$ всегда отдается предпочтение. Считаем, что вершина, имеющая меньшую оценку, с большей вероятностью окажется на оптимальном пути.

Способ, с помощью которого оценочная функция используется в процедуре ГРАФ для упорядочения вершин, можно пояснить, вновь рассмотрев пример игры в «восемь». Возьмем простую оценочную функцию

$$f(b_i)= d(b_i) + w(b_i), \qquad (6.1)$$

где $d(b_i)$ - глубина вершины b_i на дереве поиска и $w(b_i)$ - число находящихся не на нужном месте клеток в состоянии, связанном с вершиной b_i . Таким образом, конфигурация исходной вершины имеет $f(b_0) = 0+4=4$.

Результаты применения процедуры ГРАФ к игре в «восемь» с использованием такой оценочной функции приведены на рис 6.5. Значения $f(b_i)$ для каждой вершины обведены прямоугольником; отдельно выписанные числа указывают на порядок, в котором раскрываются вершины. Здесь найден тот же решающий путь, что и помощью других методов поиска, но использование оценочной функции позволило раскрыть значительно меньше вершин. Если применить оценочную функцию, равную просто $f(b_i)= d(b_i)$, то получаем процесс поиска в ширину.

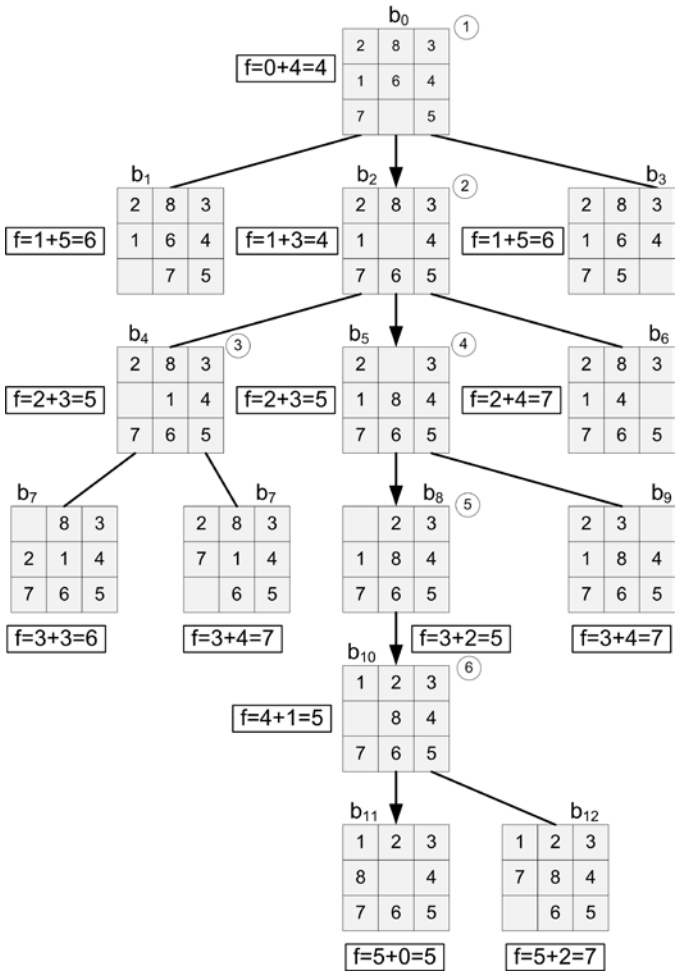

Рис 6.5 Поиск с применением оценочной функции

Выбор оценочной функции в значительной степени определяет результаты поиска. Использование оценочной функции, которая не может различить среди нескольких вершин действительно перспективную, может дать пути, стоимости которых превышают минимальную; в то же время использование функции, переоценивающей перспективность всех вершин, приводит к раскрытию слишком многих вершин.