Тема 3. ИСЧЕРПЫВАЮЩИЙ ПОИСК

3.1. Поиск с возвратом

3.1.3. Повышение эффективности поиска с возвратом

Практически все методы повышения эффективности поиска с возвратом связаны с сокращением перебора вариантов. Рассмотрим некоторые из них.

Поиск с ограничениями. Этот метод иногда называют отсечением ветвей, поскольку при этом удаляются поддеревья из дерева поиска. Основой метода является детальный анализ задачи для выявления ограничений, сокращающих процесс поиска. Практически этот метод нами использован в алгоритме решения задачи о ферзях, в котором учитываются ограничения, связанные с тем, что ферзь атакует все поля в своей строке, своем столбце и диагоналях. Оценим, как влияют эти ограничения на процесс поиска. Если нет никаких ограничений, то на доске размером $n \times n$ существует $\binom{n^2}{n}$ (для $n = 8$ около $4{,}4 \cdot 10^9$) возможных способов расстановки n ферзей. Тот факт, что в каждом столбце может находиться только один ферзь, дает n^n расстановок (для $n = 8$ около $1{,}7 \cdot 10^7$). То, что в строку можно поставить только одного ферзя, говорит о том, что вектор $(a_1, \ldots, a_n)$ может быть решением только тогда, когда он является перестановкой элементов $(1, 2, ..., n)$, что дает $n!$ возможных расстановок (для $n = 8$ около $4{,}0 \cdot 10^4$). Требование, что на диагонали может находиться только один ферзь, еще больше сокращает число возможных расстановок. Для последнего ограничения аналитическое выражение, позволяющее оценить число возможных расстановок, получить трудно, поэтому необходима экспериментальная оценка размеров дерева поиска. Например, для $n = 8$ дерево поиска содержит только 2056 вершин. Таким образом, рассмотренные ограничения позволили исключить из рассмотрения большое число возможных расстановок n ферзей, а алгоритм 3.3 исследует дерево, состоящее из 2056 вершин.

Эквивалентность решений. Метод заключается в том, что определяются критерии эквивалентности решений и процедуры перевода одного решения в другое эквивалентное решение. Если будут найдены все попарно неэквивалентные решения, то можно построить все множество решений. В задаче о ферзях два решения можно считать эквивалентными, если одно из них переводится в другое с помощью ряда вращений и/или отражений. Например, поскольку ферзь, расположенный в любом углу доски, атакует все другие угловые поля, то решений, в которых ферзи стоят более чем в одном углу, нет. Следовательно, любое решение с ферзем в позиции (1, 1) может быть переведено вращением и/или отражением в эквивалентное, в котором позиция (1, 1) пуста. Таким образом, процесс поиска можно начать с $a_1 = 2$ и получить все неэквивалентные решения. Этим сразу обрывается у дерева большое поддерево (в дереве остается 1829 вершин).

Слияние (склеивание) ветвей. Идея состоит в следующем: если два или более поддеревьев данного дерева поиска изоморфны, то достаточно исследовать только одно из них. В задаче о ферзях можно использовать склеивание, учитывая то, что если $a_1 > \lceil n/2 \rceil$, то найденное решение можно отразить и получить решение, для которого $a_1 \le \lceil n/2 \rceil$. Следовательно, поддеревья, соответствующие, например, случаям $a_1 = 2$ и $a_1 = n - 1$, изоморфны. В результате, не исследуя поддеревья, соответствующие случаям $a_1 = \lceil n/2 \rceil + 1$, $a_1 = \lceil n/2 \rceil + 2$, …, $a_1 = n$, можно найти все неэквивалентные решения. Кроме того, поскольку использование такого склеивания не влияет на ограничение в позиции (1, 1), достаточно найти решения только для $2 \le a_1 \le \lceil n/2 \rceil$. Если величина n нечетна и $a_1 = \lceil n/2 \rceil$, то можно ограничиться случаем $a_2 \le \lceil n/2 \rceil - 2$ в соответствии с тем же самым принципом. (На самом деле, если n нечетно, лучше ограничиться случаем $2 \le a_1 \le \lfloor n/2 \rfloor$ [21].) Это позволяет существенно сократить размеры дерева. Например, для $n = 8$ дерево сводится всего к 801 вершине.

Переупорядочение поиска. Данный метод может оказаться полезным, когда существование решения вызывает сомнения или когда вместо всех решений нужно найти только одно. Есть несколько способов его использования. Во-первых, если интуиция подсказывает, что решения будут иметь некоторый определенный вид, то поиск разумно построить так, чтобы раньше других исследовались возможные решения именно этого вида. Во-вторых, если это возможно, дерево следует перестроить так, чтобы вершины меньшей степени (т. е. имеющие относительно мало сыновей) были близки к корню дерева. Например, дерево поиска на рис. 3.2, *а* предпочтительнее дерева на рис. 3.2, *б*.

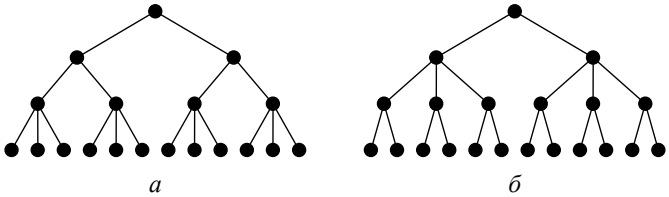

Рис. 3.2. Варианты дерева поиска:
а – предпочтительное дерево; *б* – произвольное дерево

Перестройка дерева может оказаться полезной по следующим причинам. В общем случае, чтобы обнаружить, что путь не может вести к решению, должно быть накоплено несколько ограничений, и обычно это случается на фиксированной глубине дерева. В результате большее число вершин будет запрещено, если большая часть ветвлений будет находиться ближе к листьям, чем к корню. Нужно помнить, что такой вид перестройки дерева может оказаться бесполезным, если должно быть исследовано все дерево. Кроме того, даже если не требуется исследовать все дерево, перестройка может переместить решения не ближе к началу поиска, а дальше от него.

Декомпозиция. Метод декомпозиции заключается в разложении задачи на k подзадач, решении этих подзадач и затем композиции решений подзадач в решение исходной задачи. В этом случае требования к объему памяти могут быть очень высокими, поскольку нужно хранить все решения подзадач, однако, скорость может значительно возрасти. Например, если для решения задачи размера n требуется время $c2^n$, то использование метода декомпозиции позволит уменьшить время до $kc2^{n/k}+ T$, где T – время, требуемое для композиции решений подзадач.