1.<u>Организация перебора вариантов.</u> Дерево вариантов. Способы обхода. Способы отсечений бесперспективных вариантов.

Дерево вариантов

Дерево решений представляет собой разбиение задачи P_0 на $P_1, P_2, \dots, P_k$. $\bigcup_{i=1}^{\infty} P_i = P_0$. Решить подзадачу значит найти оптимальное решение; показать, что значение оптимального решения подзадачи хуже, чем лучшее из найденных решений; показать, что задача является недопустимой.

Способы обхода

1) *Фронтальное ветвление*

Исходная задача разбивается на подзадачи $P_1, P_2, \dots, P_k$, образуя фронт ветвления. Из этих подзадач выбирается наиболее перспективная подзадача и разбивается на подзадачи, увеличивая количество вершин.

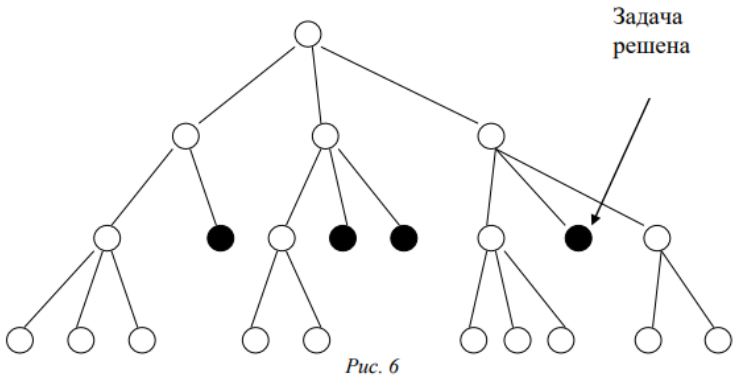
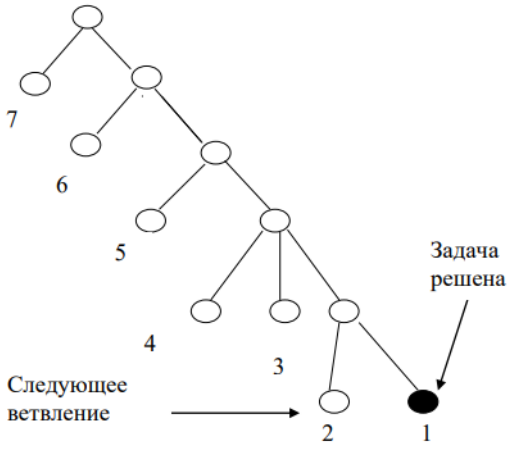


Рис. 6

2) *Односторонне ветвление*

При этом типе поиска ветвление осуществляется в последней выбранной подзадаче, и такой процесс ветвления продолжается до тех пор, пока не будет порождена подзадача, которую можно решить. В этом месте делается возврат, т. е. берется предпоследняя порожденная подзадача и ветвление продолжается в соответствующей вершине.



При этом типе поиска задачи, получаемые на каждом этапе, хранятся в стеке вместе с исходной задачей. Вновь получаемые задачи помещаются в стек, а когда подзадача разрешена, она удаляется из стека. Вид дерева решений при этом типе поиска, когда разрешается первая подзадача, показан на рисунке, где порядок приоритета исследования получаемых подзадач отражен соответствующей нумерацией.

Способы отсечений бесперспективных вариантов.

1. Отсев по повторению.
2. Отсев по допустимости.
3. Отсев по рекорду.

1. Применяется для того, чтобы исключить повторяющиеся подзадачи. Одним из примеров является поиск множества всех клик графа. Одним из наиболее простых и используемых способов отсева по повторению является построение подзадач в лексикографическом порядке, который гарантирует однозначность решений. Однако такой подход не всегда сочетается с другими способами отсечения.
2. Применяется тогда, когда мы пришли в некоторое множество и выяснили, что нам ничего допустимого не дает, то незачем в дальнейшем туда ходить. Примером являтся задача определения множества всех контуров ориентированного графа.
3. Применяется если заранее можно определить, что некоторая подзадача не даст решения лучше, чем построенное алгоритмом ранее (рекорд), то незачем решать такую подзадачу. При организации перебора вариантов решения строится дерево вариантов, причем частичным решениям соответствуют внутренние вершины дерева, а решениям исходной задачи – листья. Целью организации поиска является такой обход древа, который позволяет построить все оптимальные решения, просмотрев минимальное число вершин дерева. Самое простое решение поставленной задачи это полный обход дерева. Однако существуют механизмы, позволяющие для многих задач существенно сократить число рассматриваемых вершин. Для этого достаточно определить, какие из подзадач целесообразно рассматривать (ветвить), а какие являются неперспективными и их можно игнорировать в силу того факта, что они заведомо не приведут к построению оптимального решения.

2.Нижняя (верхняя) оценки значения оптимального решения. Способы их нахождения.

В отсеве по рекорду используются оценки значения оптимального решения.

Для вычисления оценки, существует несколько способов. Часто рассматривается более общая задача, которая легко решается. Для этого, например, от дискретных переменных переходят к непрерывным, либо убирают (ослабляют) некоторые ограничения, тем самым гарантируя, что значение оптимального решения более общей задачи не больше в задаче минимизации и не меньше в задаче максимизации, чем значение оптимального решения исходной задачи. Затем значение оптимальное решение более общей задачи берется в качестве оценки.
Если в задаче минимизации для вершины дерева x выполняется неравенство

$$F < F(x) + \text{НО}(\bar{x}),$$

то понятно, что

$$F < F(x) + opt(\bar{x})$$

поэтому частичное решение x не приведет нас к оптимальному решению. Следовательно, для вершины x можно производить отсечение по рекорду. Очевидно, что в качестве нижней оценки в задаче минимизации можно всегда взять $\text{НО}(\bar{x}) = 0$. В этом случае отсечение частичного решения по рекорду происходит только в случае, если значение целевой функции для частичного решения больше значения рекорда.

Аналогично, если в задаче максимизации для вершины дерева x выполняется неравенство

$$F > F(x) + \text{ВО}(\bar{x}), \text{то}$$

$$F > F(x) + opt(\bar{x}),$$

следовательно частичное решение x не приведет нас к оптимальному решению и можно выполнить отсечение вершины x по рекорду. В качестве верхней оценки в задаче максимизации можно взять $BO(\bar{x}) = +\infty$. В этом случае отсечения частичного решения по рекорду не происходят никогда, происходит только пересчет рекорда, следовательно выполняется полный обход дерева решений.

Метод организации перебора вариантов решения с отсечениями по рекорду часто называют методом «ветвей и границ».

Теорема 1 Если нижняя оценка подзадачи совпадает со значением ее оптимального решения, т. е.

$$\text{НО}(x) = opt(\bar{x}),$$

то это позволяет генерировать для задачи минимизации только оптимальные решения, отсекая в дереве все неперспективные частичные решения.

Аналогично, для задачи максимизации справедлива следующая теорема.

Теорема 2 Если верхняя оценка подзадачи совпадает со значением ее оптимального решения, т. е.

$$\text{ВО}(x) = opt(\bar{x}),$$

то это позволяет генерировать для задачи максимизации только оптимальные решения, отсекая в дереве все неперспективные частичные решения.

3. Относительная погрешность и гарантированная оценка алгоритмов. Пример.

Относительная погрешность алгоритма A на входном наборе I определяется как величина:

$$\frac{|F(x^{\text{опт}}) - F(x^A)|}{|F(x^{\text{опт}})|}$$

Иногда для оценки качества алгоритма A рассматривают функцию:

$$\Delta_A = \frac{\inf F(x^A)}{|F(x^{\text{опт}})|}$$

где inf берется по всем возможным наборам входных данных.

Под гарантированной оценкой точности решения, получаемого при помощи алгоритма A, понимаем:

- Для задачи на максимум это будет нижняя оценка величины Δ_A

$$z \leq \Delta_A = \frac{|F(x^A)|}{|F(x^{\text{опт}})|} \leq 1$$

- Для задачи на минимум это верхняя оценка

$$1 \leq \Delta_A = \frac{|F(x^A)|}{|F(x^{\text{опт}})|} \leq z$$

Пример:

Пусть $p_1, p_2, \dots, p_n$ −длительности программ. Имеются два носителя одинаковой длины α. Необходимо вместить как можно больше программ на носители.

Поскольку $F(x^{\text{отн}})$ не известно, то будем оценивать $\frac{F(x^A)}{BO(I)}$ (максимум) и $\frac{F(x^A)}{HO(I)}$ (минимум). В этом случак $\frac{F(x^A)}{F(x^{\text{опт}})} \geq \frac{F(x^A)}{BO(I)}$. Для минимума аналогично.

Предположим, что $\exists k$: $\sum_{i=1}^{k} p_k \leq 2\alpha$ и $\sum_{i=1}^{k+1} p_k \geq 2\alpha$, тогда $F(x^{\text{отн}}) \geq k$. Предположим, что существует алгоритм A, для которого $F(x^A) \geq k - 1$, тогда A – приближенный алгоритм с относительной погрешностью $|F(x^{\text{отн}}) - F(x^A)| \leq |k - (k-1)| = 1$.

4. *Онлайн версии задач. Способы оценки качества онлайн алгоритмов. Нижняя граница для онлайн алгоритма. Пример.*

Рассмотрим задачу об упаковке в контейнеры. Есть $a_1, a_2, \dots, a_m$ контейнеров и $L_1, L_2, \dots, L_n$ предметов веса $C \in [0; 1]$. Нужно распределить предметы по контейнерам так, чтобы вес предемтов в контейнерах не более 1 и $m \to min$.

Существует offline и online версии решения. Рассмотрим основные из них:

1. FF-первый подходящий
   Берем предмет и пытаемся найти ему место (самое первое, куда он поместится)
   $$\frac{F(x^{FF})}{F(x^{\text{опт}})} \leq 1{,}7(\text{отличается не более, чем на } 70\%)$$
2. BF – лучший подходящий.
   Берем предметы не по порядку, а по невозрастающей последовательности (каждый предмет помещаем в самый загруженный контейнер)
   $$\frac{F(x^{FF})}{F(x^{\text{опт}})} \leq 1{,}7$$
   Недостаток: Пока поступают элементы, все контейнеры являются открытыми.

5. Что такое (быстрые) ε-приближенные алгоритмы. Пример

PTAS это семейство алгоритмов, зависящих от параметра ε , таких, что для произвольного набора данных некоторой оптимизационной задачи и параметра $\varepsilon > 0$ алгоритм семейства за полиномиальное время находит решение с целевой функцией $S < (1 + \varepsilon)OPT$, где OPT минимум целевой функции. Например, для задачи коммивояжера в Евклидовом пространстве существует PTAS, которое находит путь обхода длины не более $(1 + \varepsilon)L$, где L длина кратчайшего пути.

Время выполнения PTAS должно полиномиально зависеть от n при любом фиксированном ε, но может произвольно меняться при изменении ε.

Алгоритмы со временем выполнения $O\ (n^{1/\varepsilon})$ или даже $O(n^{exp\left(\frac{1}{\varepsilon}\right)})$ являются алгоритмами PTAS.

В алгоритмах PTAS показатель степени в оценке сложности может расти катастрофически при убывании ε, например, когда время выполнения $O(n^{\left(\frac{1}{\varepsilon}\right)!})$, что делает этот класс алгоритмов малоинтересным с практической точки зрения. Можно определить **эффективную приближенную схему полиномиального времени** или **efficient polynomial-time approximation scheme** (**EPTAS**), для которой время выполнения должно быть $O(n^c)$, где константа c не зависит от ε. Это гарантирует, что увеличение размера входных данных увеличивает время выполнения независимо от величины ε; однако множитель под знаком O при этом продолжает произвольно зависеть от ε. Дальнейшим ограничением более полезным на практике является **приближенная схема полностью полиномиального времени** или **fully polynomial-time approximation scheme** (**FPTAS**), которая требует, чтобы время выполнения алгоритма полиномиально зависело и от размера задачи *n*, и от 1/ε. Примером задачи для которой существует FPTAS является задача о ранце или задача о контейнерах, условие которой была написано в предыдущем номере.

6 Предложить приближенный алгоритм с гарантированной оценкой для следующей задачи:

$$C_1x_1 + \dots + C_nx_n \rightarrow min$$

$$A_1x_1 + \dots + A_nx_n \geq B$$

$$X_i = 0 \text{ или } 1, \quad i = 1, \dots, n$$

Задача состоит в том, чтобы найти подмножество предметов так, чтобы сумма их размеров была не меньше константы, когда сумма затрат была минимально.

Для начала отсортируем $\frac{C_1}{A_1} \leq \frac{C_2}{A_2} \leq \dots \leq \frac{C_n}{A_n}$

Выберем такое k_1, что $\sum_{i=1}^{k_1} A_i < B < \sum_{i=1}^{k_1+1} A_i$

Допустим, что $S = (A_1, A_2, \dots, A_{k_1+1})$ – возможное решение

Пусть $S_1 = (A_1, \dots, A_{k_1})$

$$S = S_1 \cup \{A_{k_1+1}\}$$

Пусть $k_1 + 1, k_1 + 2, \dots, k_2 - 1$ –ряд индексов (возможно пустой), таких что для всех соответствующих $j \in \{k_1 + 2, k_1 + 3, \dots, k_2 - 1\}$ выполняется $\sum_{i=1}^{k_1} A_i + A_j \geq B$

Пусть $L_1 = (A_{k_1+1}, \dots, A_{k_2-1})$, тогда все $S_1 \cup \{A_j\}, j \in \{k_1 + 2, \dots, k_2 - 1\}$ – тоже может быть решением.

Пусть существует k_2, что $\sum_{i=1}^{k_j} A_i + A_{k_2} < B; k_3 \geq k_2 \Rightarrow$

$\sum_{i=1}^{k_1} A_i + \sum_{i=k_2}^{k_2} A_i < B < \sum_{i=1}^{k_j} A_1 + \sum_{i=k_2}^{k_3+1} A_i$

$$S_2 = (A_{k_2}, A_{k_2+1}, \dots, A_{k_3})$$

Откуда $S_1 \cup S_2 \cup \{A_j\}$ –возможное оптимальное решение.

Теперь из всехвозможных оптимальных решений выбираем то, где сумма $C_1x_1 + \dots + C_nx_n$ –минимальна.