Модуль 3. NP-трудные задачи

Лекция 13

Точные алгоритмы:
метод ветвей и границ,
динамическое программирование.

План лекции

- Метод ветвей и границ
  - Общий принцип
  - МВГ для задачи коммивояжёра
    - ✓ Варианты ветвления
    - ✓ Варианты оценки
- Динамическое программирование
  - Алгоритм Беллмана-Хелда-Карпа

Метод ветвей и границ

Метод ветвей и границ

- Предназначен в первую очередь для решения NP-трудных оптимизационных задач.
- Сочетание двух операций:
  - ✓ Ветвление.
  - ✓ Оценка верхней/нижней границы и «отсечение» неперспективной ветви.

Метод ветвей и границ

Ветвление

- Всё множество допустимых решений S(x) последовательно *виртуально* делится на подмножества:

$$S = \bigcup S_i$$

- Подмножества рекурсивно делятся дальше:

$$S_i = \bigcup S_{i,j}$$

- Для разделения множества решений на подмножества обычно последовательно накладывают условия/ограничения, которым должны удовлетворять решения из заданного подмножества.

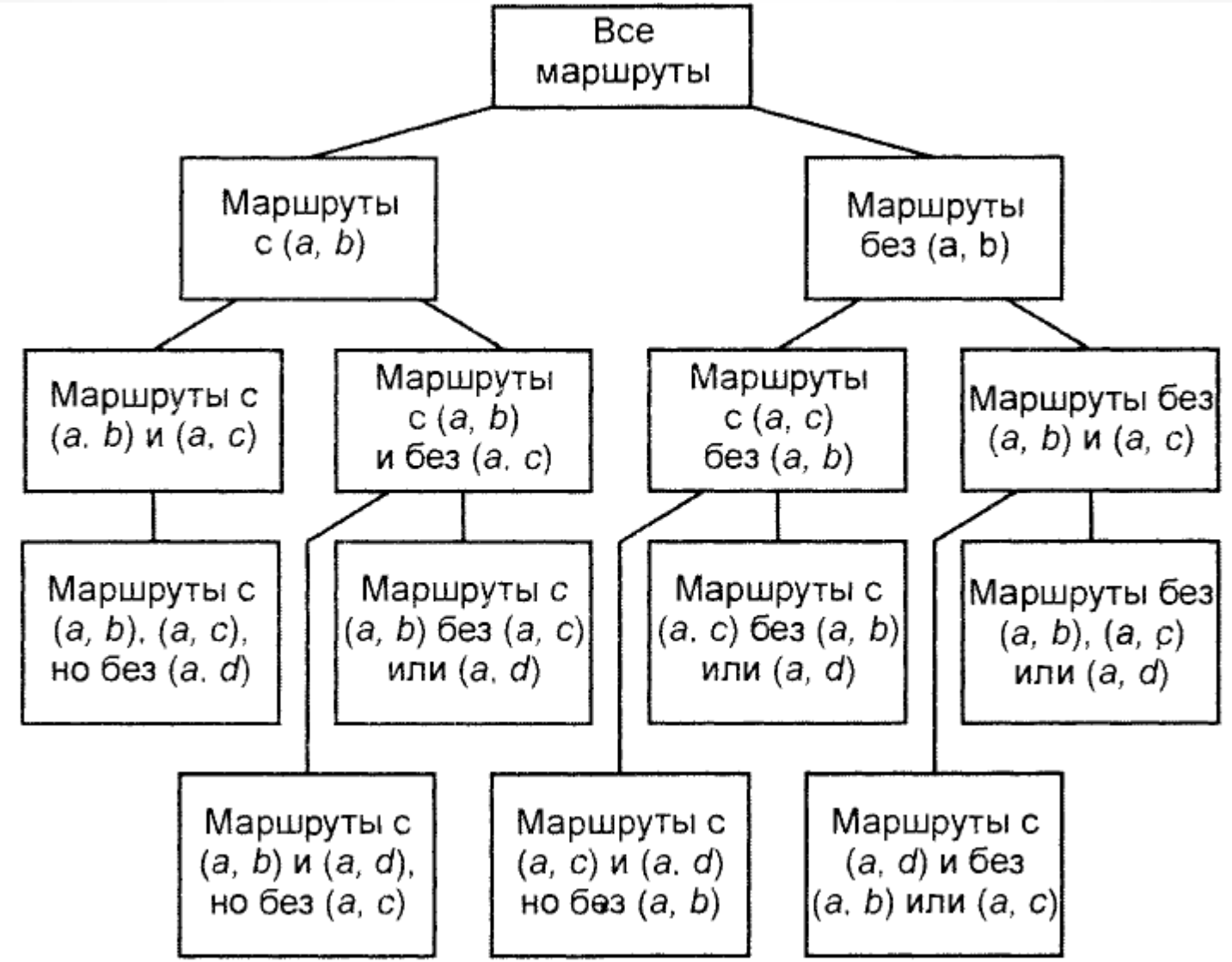
Метод ветвей и границ

Примеры ограничений для задачи коммивояжёра

- По принципу «содержит» / «не содержит» указанные дуги (алгоритм Литтла).

  Выбрать дугу $e \in E$ и разделить S(x) на два подмножества: $S_{\{e\}}(x) = \{Z: e \in Z\}$ и $S_{\{\bar{e}\}}(x) = \{Z: e \notin Z\}$. Потом выбираем другую дугу и аналогичным образом делим $S_{\{e\}}(x)$ и $S_{\{\bar{e}\}}(x)$ на более мелкие подмножества.

Метод ветвей и границ



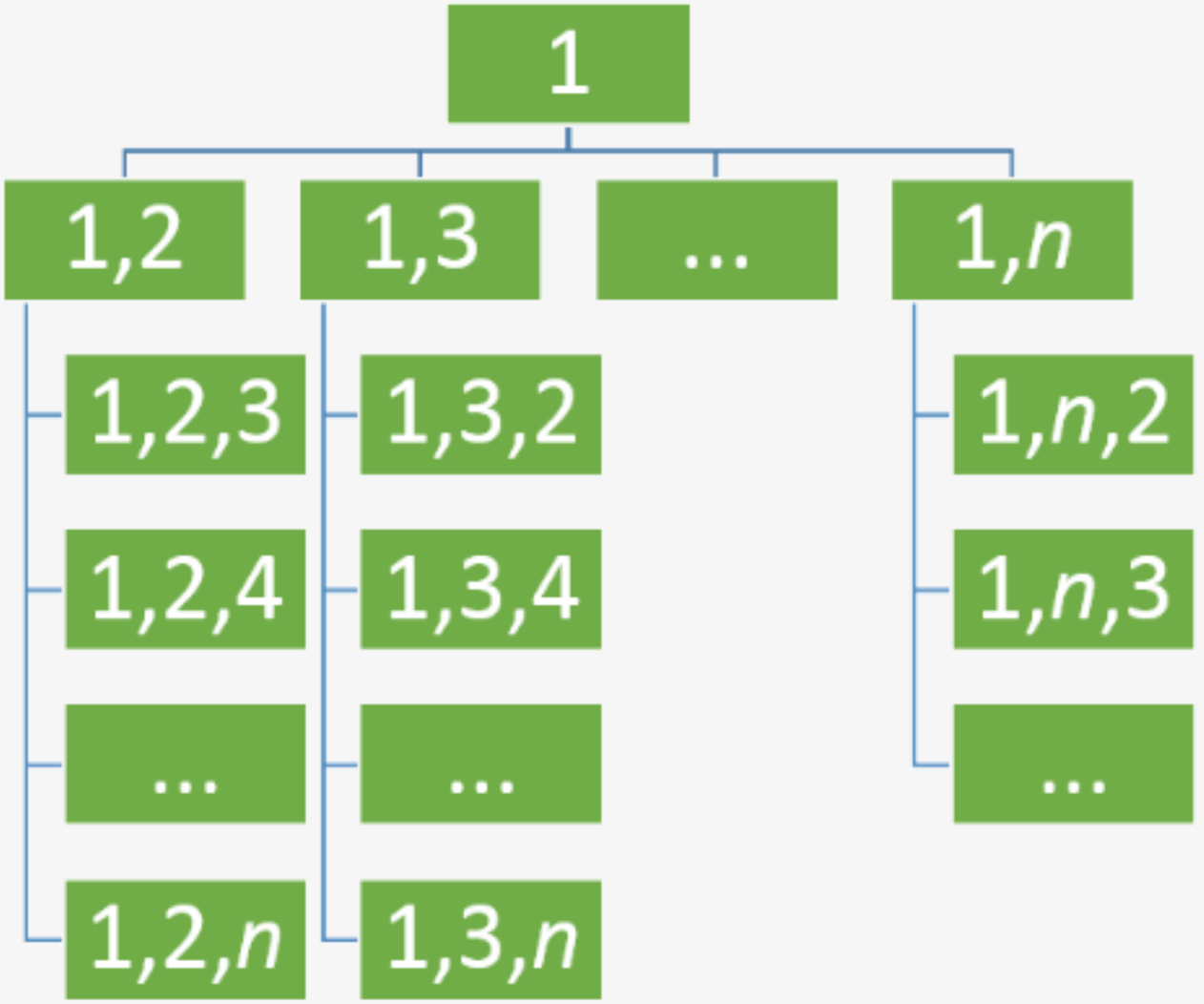
Метод ветвей и границ

Примеры ограничений для задачи коммивояжёра

- Частичный путь

  Фиксируем фрагмент цикла — какие вершины цикл проходит последовательно друг за другом. Стартовая вершина всегда фиксирована: «1». Поэтому разделяем все возможные циклы на подмножества, содержащие фрагменты: «1,2», «1,3»,..., «1,*n*». Потом делим множество «1,2»: «1,2,3», «1,2,4», …, «1,2,*n*».

Метод ветвей и границ



Метод ветвей и границ

Принцип отсечения неперспективных ветвей

- Вычисляем оценку («границу») стоимости решений, входящих в анализируемое подмножество. Для задач минимизации вычисляем оценку снизу, для максимизации — оценку сверху.
- Сравниваем оценку со стоимостью рекордного (наилучшего на данный момент) решения. Если оценка хуже (для минимизации — выше, для максимизации — меньше), чем рекордное решение — данное подмножество далее не анализируем.

Метод ветвей и границ

Важно иметь в виду: на вычисление оценки также тратится время, и важно соблюдать баланс между качеством оценки и сложностью её вычисления.

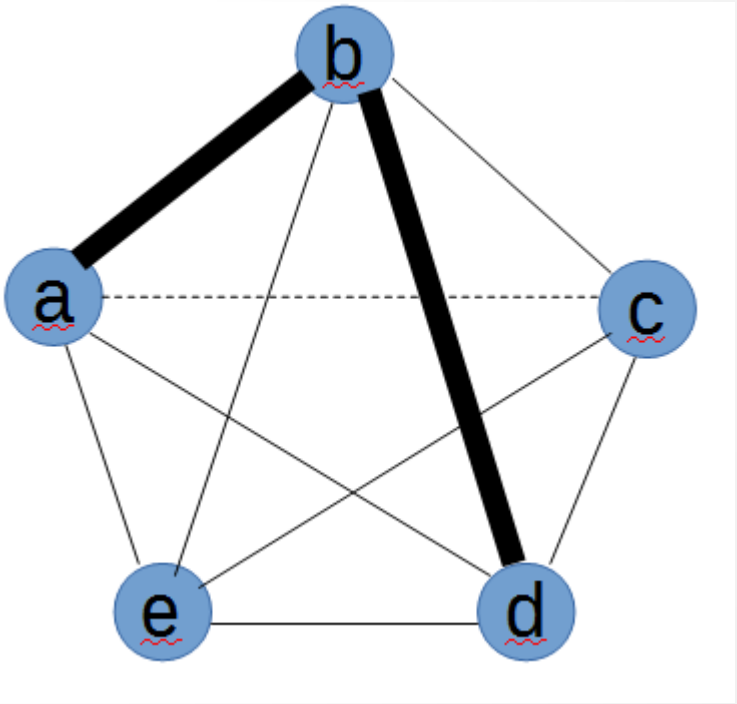
Возможный вариант: применять разные оценки на разных уровнях «дерева перебора»: на верхних уровнях более сложные, но более точные, на низших уровнях — менее точные, но более быстрые (**подумайте — почему так?**).

Метод ветвей и границ

Варианты вычисления границ (оценка снизу) для задачи коммивояжёра:

(1) Выбрать минимальную стоимость дуги (ещё не включённой в цикл), умножить на количество ещё не пройденных вершин графа и прибавить стоимости дуг, уже включённых в цикл.

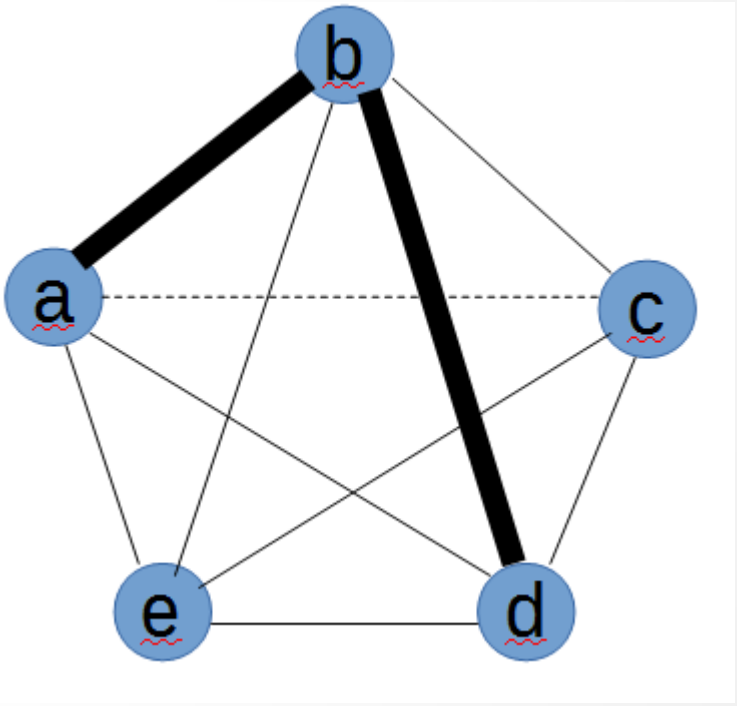
w’ = w(a,b)+w(b,d)+3*w(a,c)



Метод ветвей и границ

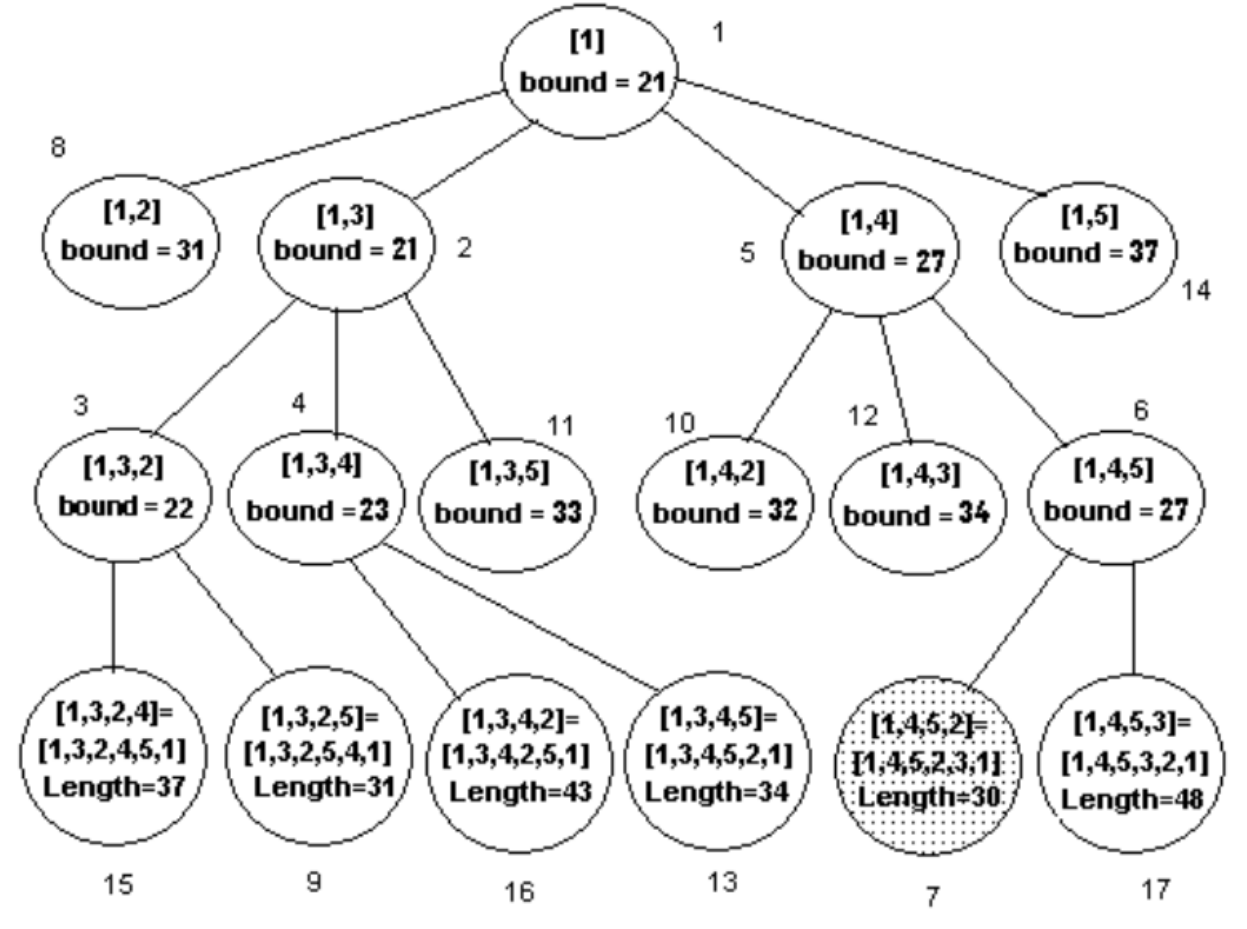
Варианты вычисления границ (оценка снизу) для задачи коммивояжёра:

(2) Для каждой вершины найти две инцидентные ей дуги наименьшей стоимости (обязательно включая дуги, уже вошедшие в цикл — если такие есть) и взять их среднее арифметическое. В качестве оценки использовать сумму этих средних



МВГ + «сначала наилучшие»

«Классический» МВГ = обход дерева решений в глубину



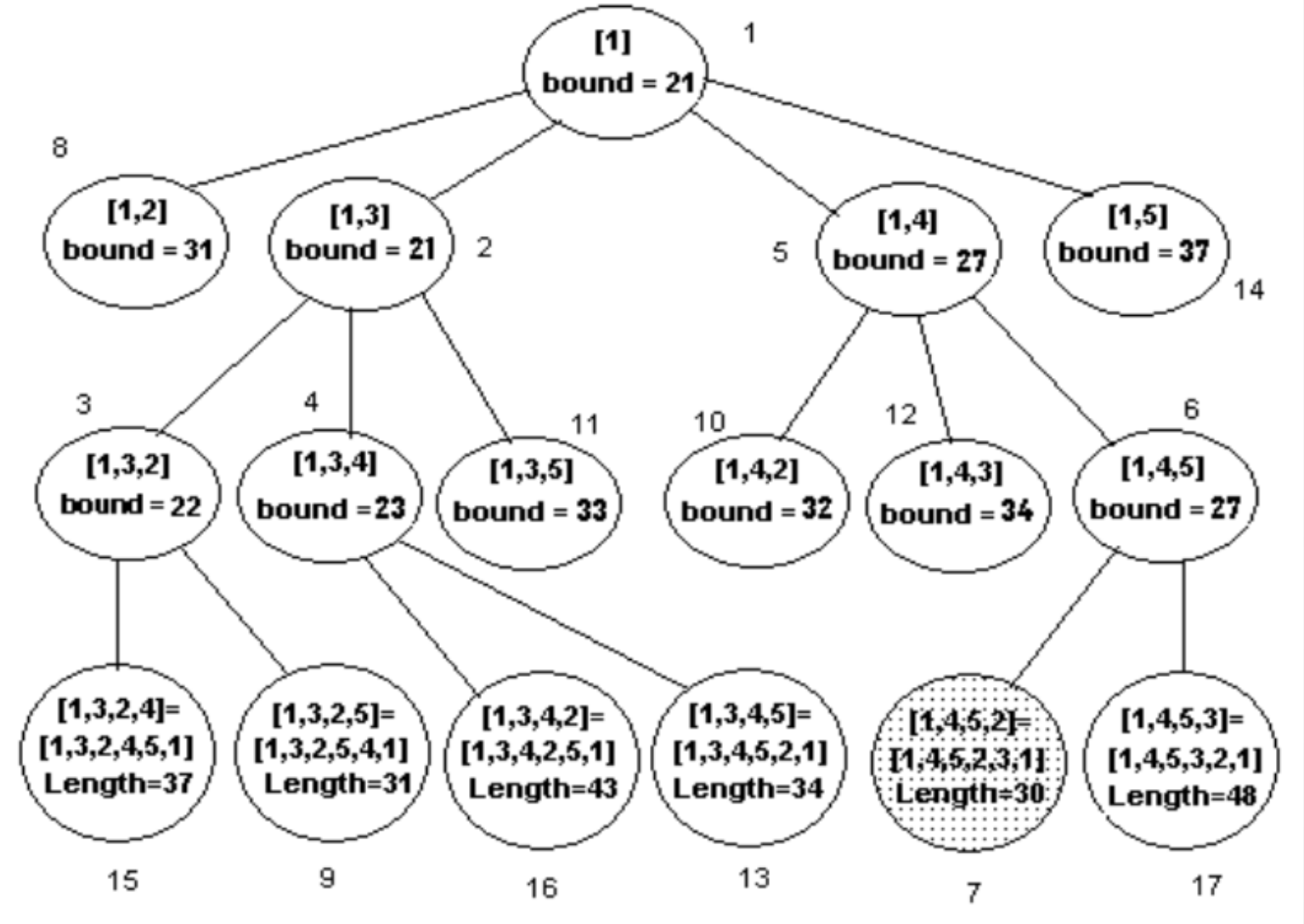
МВГ + «сначала наилучшие»

Применение эвристики «сначала наилучшие» («best-first»):

- Каждый «вариант» (подмножество решений) после расчёта границы сохраняем в очередь с приоритетами.
- Для ветвления берём не текущий (только что оцененный) вариант, а извлекаем из очереди наилучший (с минимальной/максимальной оценкой).

Метод содержит черты обхода дерева решений в ширину.

МВГ + «сначала наилучшие»



Метод ветвей и границ

Сложность метода ветвей и границ: в худшем случае – экспоненциальная, совпадает со сложностью полного перебора $O(n!)$.

Вопрос: как строить начальный цикл-рекордсмен?

1) Никак – инициализируем рекорд значением $+\infty$. 
2) Найти эвристическое (жадное?) решение.

Динамическое программирование. Алгоритм Беллмана-Хелда-Карпа

Динамическое программирование

Принципы динамического программирования:

- ✓ Исходная задача сводится к последовательности подзадач меньшего размера.
- ✓ Задачи решаются от меньшей к большей. Результаты сохраняются в таблицу.

Для корректности алгоритма ДП должен выполняться ***принцип оптимальности*** (часть оптимального решения является оптимальным решением подзадачи).

Динамическое программирование

Будем считать, что

1) Вершины графа пронумерованы.
2) Оптимальный цикл всегда начинается в вершине 1.

Предположим, что оптимальный цикл возвращается в начальную-конечную вершину 1 из вершины i ($i \in \{2, \dots, n\}$). Тогда такой цикл состоит из пути $P(V, i)$ и дуги $(i, 1)$.

Здесь через $P(U, i)(U \subseteq V)$ обозначен путь, который начинается в вершине 1, оканчивается в вершине $i \in U$ и проходит только через вершины множества U, не более одного раза через каждую.

Через $W(U, i)$ обозначим вес $P(U, i)$.

Динамическое программирование

Лемма (принцип оптимальности для ЗК).

Для любого $U \subseteq V$ и любой вершины $i \in U$, если последняя дуга в $P(U, i)$ ведёт из k в i, то часть пути $P(U, i)$ от 1 до k является $P(U \backslash \{i\}, k)$.

Это даёт идею для вычислительной процедуры:

1) Для решения задачи надо найти $i \in \{2, \dots, n\}$, для которой выражение $W(V, i) + w(i, 1)$ достигает минимума.
2) Значения $W(U, i)$ находим, вычисляя минимум $W(U \backslash \{k\}, k) + w(k, i)$ по всем $k \in U$.
3) Базовые случаи: $|U| = 2$. Для любого такого $U = \{1, i\}$ устанавливаем $W(U, i) = w(1, i)$.

Динамическое программирование

Для программной реализации необходима двумерная таблица A с $(2^{n-1}-1)$ строками и $n-1$ столбцами. Каждая строка содержит подмножество U (обязательно не пустое и содержащее 1). В ячейке таблицы с координатами (i,j) хранится величина $W(U,j)$, где множество U задаётся характеристическим вектором, представляющим двоичное разложение числа i.

Временная сложность алгоритма: $O(n^2 2^n)$.

Это экспоненциальная величина, но всё равно значительно меньше, чем $O(n!) \approx O(\frac{n^n}{e^n})$.