Модуль 2. Ускорение перебора

Лекция 6

Распространение ограничений

План лекции

- Задача удовлетворения ограничений
  - Формулировка
  - Примеры практических задач
- Методы решения
  - Полный перебор
  - Древовидный перебор (поиск с возвратами)
  - Распространение ограничений

Задача удовлетворения ограничений

- V = $\{v_1, v_2, \ldots, v_n\}$ — переменные
- Для каждой переменной v_i задана её область определения: D_i
- Пусть $S \subseteq V$ — диапазон переменных, $S = \{v_{i1}, v_{i2}, \ldots, v_{ik}\}$; $R \subseteq D_{i1} \times D_{i2} \times \ldots \times D_{ik}$ – отношение. Тогда пара C=(S, R) — ограничение
- Присваивание: f: $V \rightarrow D_1 \times D_2 \times \ldots \times D_n$
- Присваивание f удовлетворяет ограничению C=(S, R), если $(f(v_{i1}), f(v_{i2}), \ldots, f(v_{ik})) \in R$.

Задача удовлетворения ограничений

Задача «Удовлетворение ограничений» („Constraint Satisfaction Problem“, CSP): для заданных (X,D,C) найти присваивание, удовлетворяющее ограничениям.

Варианты задачи:

- Найти все допустимые присваивания
- Для каждого ограничения («предпочтения») указана его «стоимость». Найти присваивание, нарушающее множество ограничений минимальной стоимости.

Задача удовлетворения ограничений

Примеры практических (и «игрушечных») задач, представимых в виде ЗУО:

- Решение линейных уравнений (над конечными полями)
- Календарное планирование
- Вершинное покрытие
- Задача о рюкзаке
- Задача об N ферзях
- Кроссворды (обычные и японские)
- … и многие другие

Задача удовлетворения ограничений

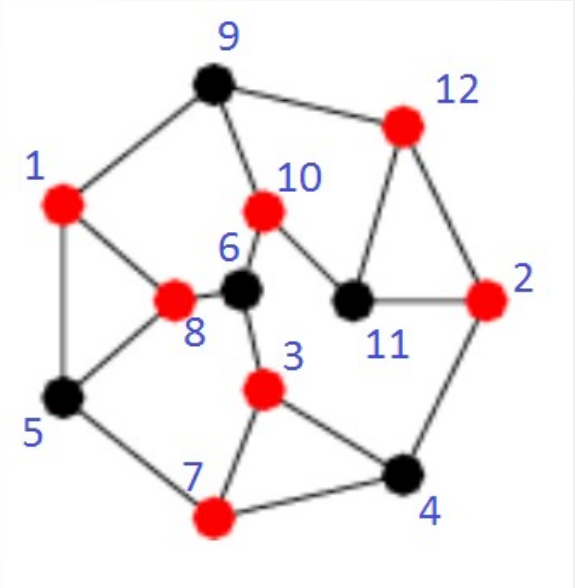
Сведение задачи «Вершинное покрытие» к ЗУО:

- Переменнные-вершины графа.
- Области: D = {0,1}

  1-“входит в покрытие“

  0-“не входит в покрытие“
- R = {(0,1), (1,0), (1,1)}
- Чтобы учесть мощность покрытия:

  AtMost(V,k) = „$\sum f(v_i) \leq k$“



Задача удовлетворения ограничений

Виды ограничений

- Унарные — можно устранить, модифицировав области определения.
- Бинарные (двоичные)
- Ограничения произвольной арности
- Агрегатные: *AtMost*, *AllDiff*
- Кумулятивные: $\sum f(x_i)w_i \leq b$

Любые ограничения можно свести к бинарным. Но это может затруднить решение задачи — для конкретных небинарных ограничений могут существовать более эффективные алгоритмы.

Задача удовлетворения ограничений

Методы решения ЗУО будем рассматривать на примере задачи о раскраске вершин графа.

- Дан граф G(V,E).
- (Допустимая) раскраска графа в d цветов — отображение f:V$\rightarrow${1,…,d}, при котором для любого ребра (u,v)$\in$V выполняется: f(u)$\neq f$(v).

Методы решения ЗУО

Самый «простой» метод — полный перебор всех возможных вариантов

- Последовательно генерировать все возможные присваивания — схема «строки в произвольном алфавите».
- Для каждого сгенерированного варианта проверять допустимость.

Сложность: $O(d^n)$, где n — количество переменных, $d=\max\{D_i\}$.

Методы решения ЗУО

Недостаток: проверяет все варианты, не отсекая заведомо неперспективные.

(…, 1,1,1,…)

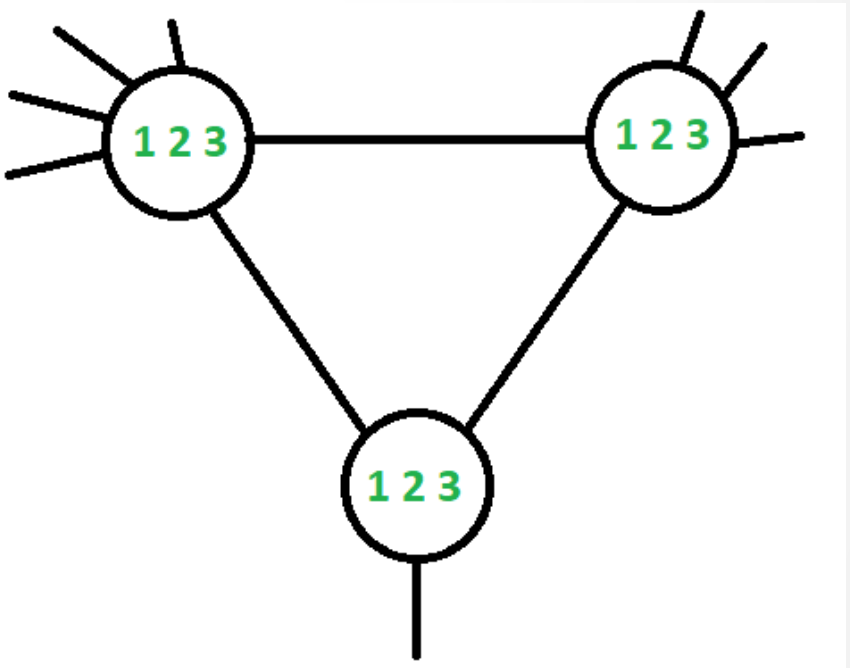
(…, 1,1,2,…)

(…, 1,1,3,…)

…

(…, 2,1,1,…)

…

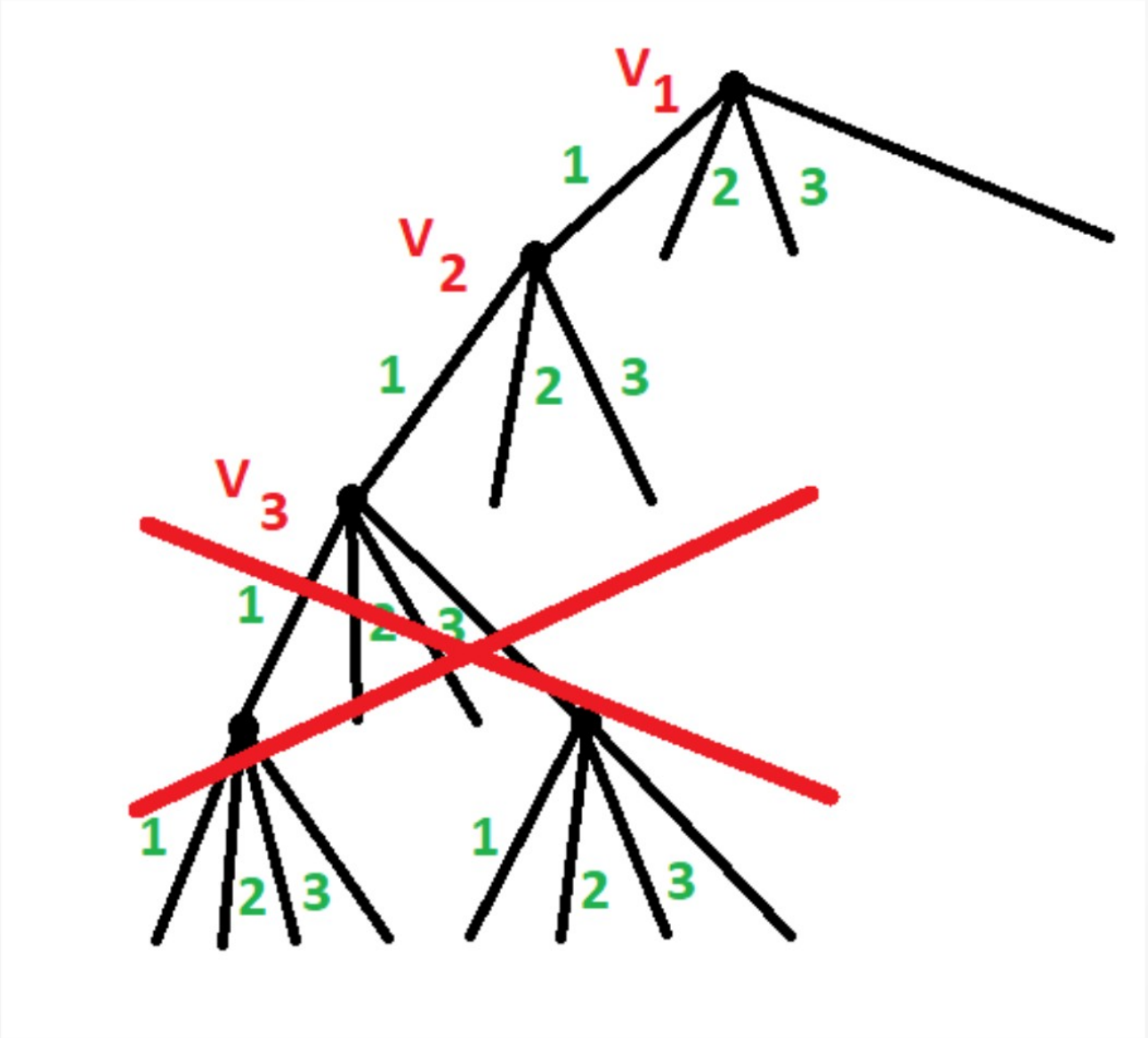


Методы решения ЗУО

Древовидный перебор (поиск с возвратами).

- Присваиваем значения переменным последовательно (например, в порядке следования: v_1, v_2, …, v_n).
- Для каждого частичного присваивания проверяем выполнение ограничений, в которых все входящие переменные уже получили значения.
- Если хотя бы одно ограничение нарушено - «отсекаем ветвь», т. е. прекращаем дальнейшее присваивание и возвращаемся к предыдущей переменной (присваиваем ей другое значение).

Методы решения ЗУО



Методы решения ЗУО

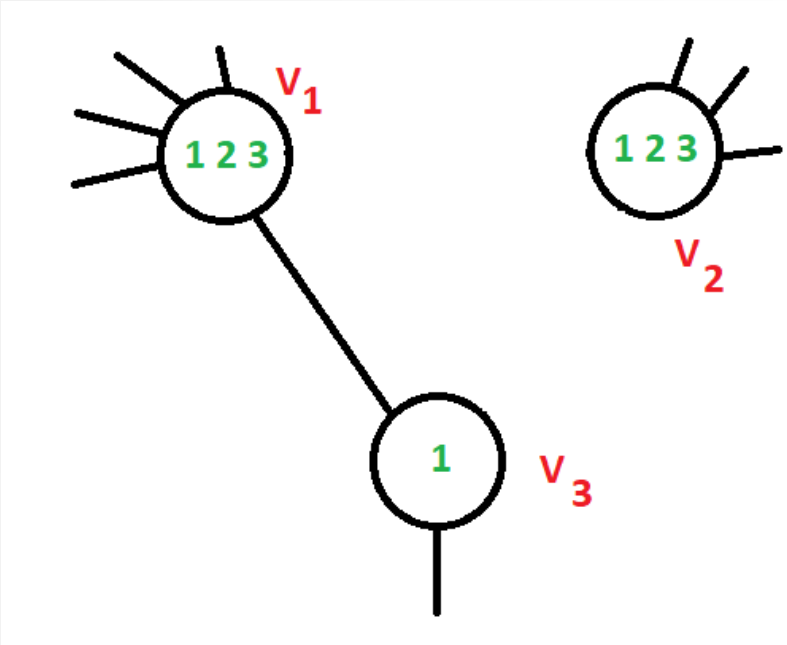
Древовидный перебор позволяет сократить количество вычислений. Но и его можно ускорить, проводя по ходу перебора более тонкий анализ и применяя ускоряющие эвристики.

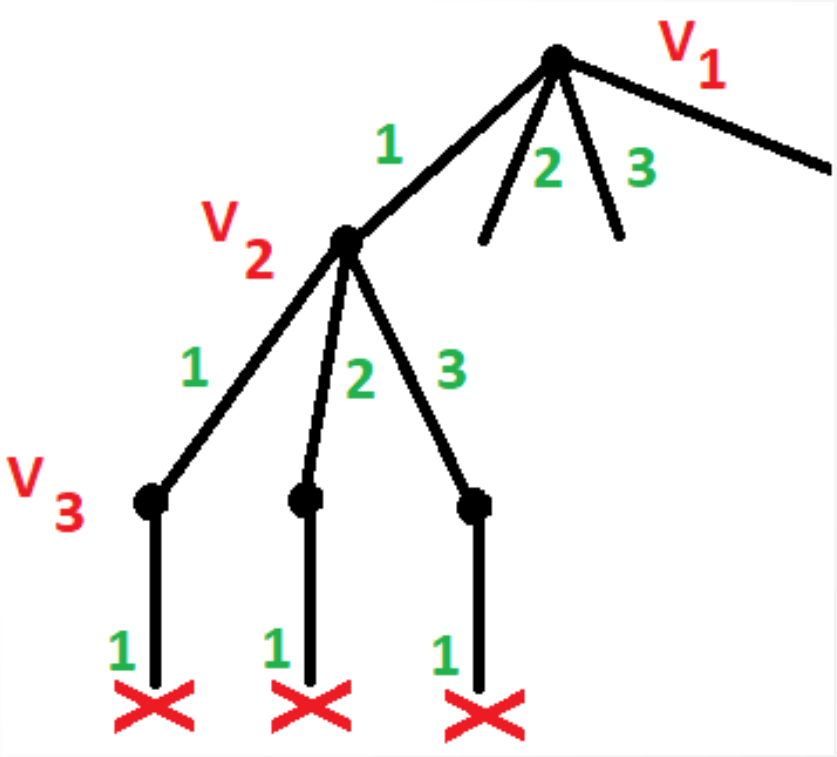
- Интеллектуальный переход при возврате.
- Гибкий учёт ограничений для ещё неприсвоенных переменных в зависимости от значений присвоенных переменных.
- Эвристически определять порядок, в котором рассматриваются переменные, и порядок проверки значений для конкретной переменной.

Методы решения ЗУО

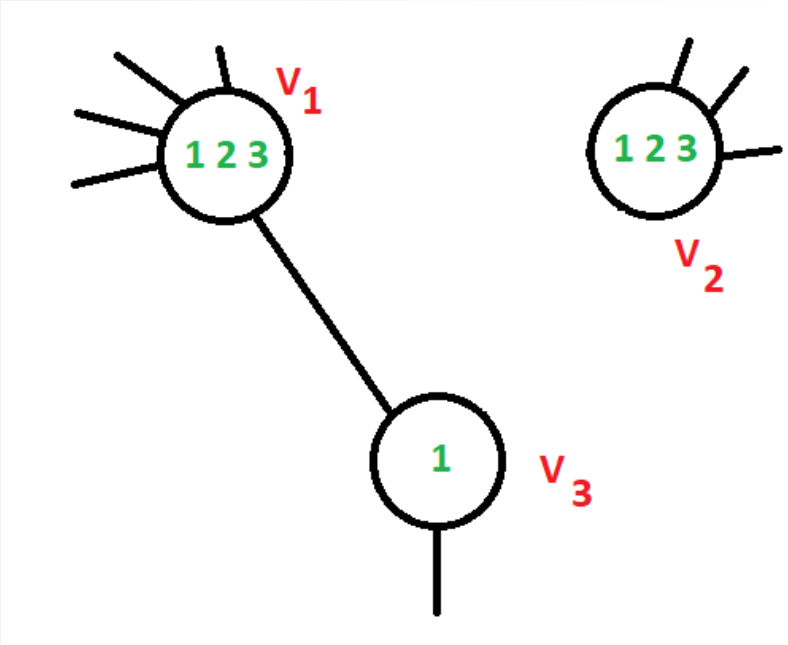
Основная идея интеллектуального перехода при возврате: если встретили нарушение ограничения и планируем возвращаться к ранее рассмотренным вариантам, то анализируем причину нарушения и переходим к тому узлу дерева перебора, из-за которого возник конфликт.

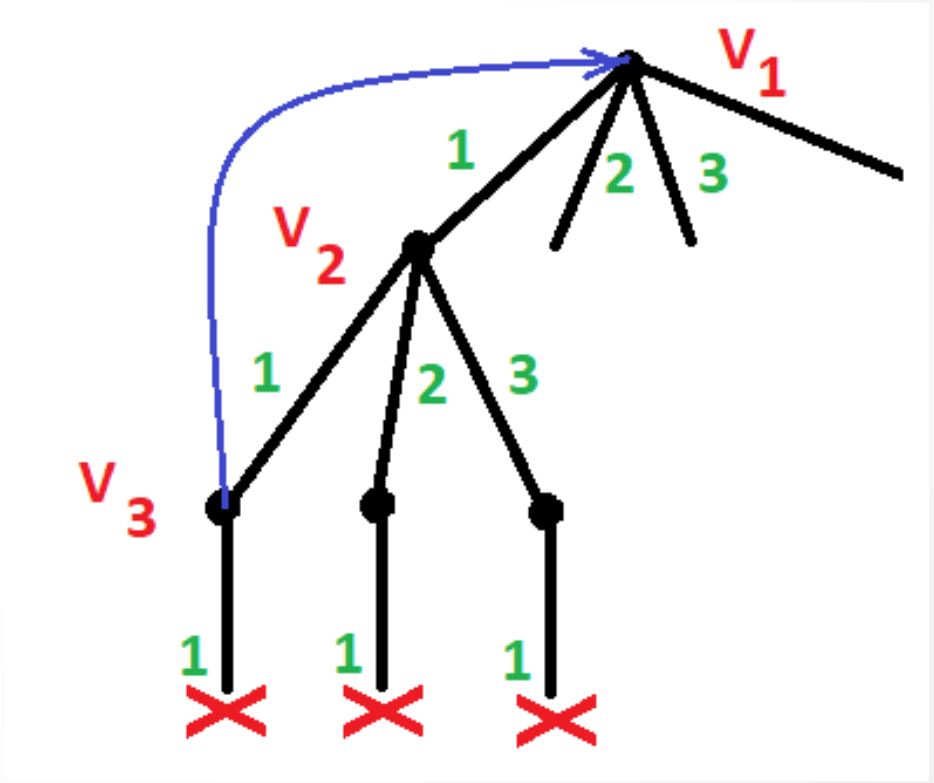
Методы решения ЗУО





Методы решения ЗУО





Методы решения ЗУО

Гибкий учёт ограничений при переборе.

Рассмотрим бинарную ЗУО.

Введём граф ограничений G(V,E):

- Вершины — переменные ЗУО.
- Две вершины связаны ребром, если есть ограничение, в которое входят обе эти вершины.

Методы решения ЗУО

Граф является *совместным по вершинам*, если все переменные ЗУО (вершины графа) удовлетворяют унарным ограничениям.

Если граф ограничений не совместен по вершинам, то при переборе проверяются лишние (заведомо неподходящие) варианты присваиваний.

Для достижения совместности по вершинам, надо удалить из области определения каждой переменной те значения, которые нарушают какое-либо унарное ограничение.

Сложность: O(*dn*).

Методы решения ЗУО

Совместность по дугам.

Пусть (u,v) — ребро на графе ограничений.

Дуга (u,v) называется *совместной*, если для для каждого *x* из D_u найдётся *y* из D_v, при котором присваивание (u=x,v=y) удовлетворяет ограничениям на (u,v).

Важно: граф ограничений — неориентированный, но совместность — ориентированное отношение. Из совместности (u,v) не следует совместность (v,u).

Граф ограничений называется *совместным по дугам*, если все его дуги совместны.

Методы решения ЗУО

Совместность по дугам.

Пусть (u,v) — ребро на графе ограничений.

Дуга (u,v) называется *совместной*, если для для каждого *x* из D_u найдётся *y* из D_v, при котором присваивание (u=x,v=y) удовлетворяет ограничениям на (u,v).

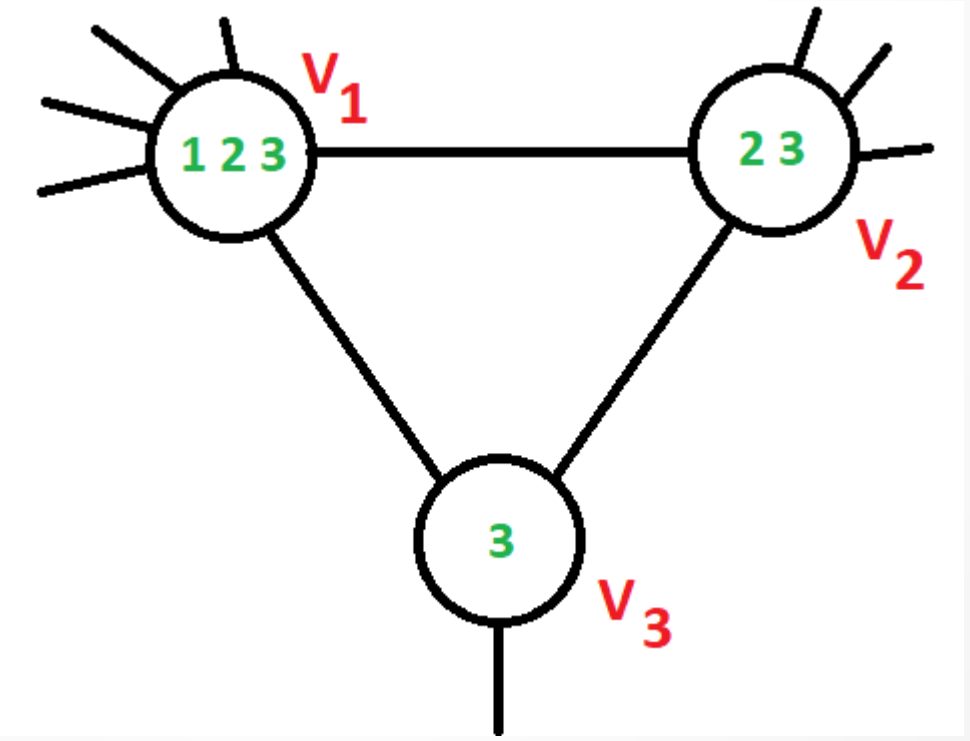
Важно: граф ограничений — неориентированный, но совместность — ориентированное отношение. Из совместности (u,v) не следует совместность (v,u).

Граф ограничений называется *совместным по дугам*, если все его дуги совместны.

Методы решения ЗУО

Дуга (3,2) совместна.

Дуга (2,3) — несовместна.



Методы решения ЗУО

Алгоритм достижения дуговой совместности.

Многократно (пока есть необходимость) проверяем/модифицируем области определения каждой переменной, проверяя на совместность инцидентных дуг.

Процедура Revise(u,v)

FlagDel := False;

Для каждого x из D_u выполнить:

{

Если не существует y из D_v, при котором выполяется ограничение на (u,v), то

{

Исключить x из D_u;

FlagDel := True;

}

}

Вернуть FlagDel.

Методы решения ЗУО

Процедура AC-3(G)

```
    Q :=  E;
    Пока Q не пусто:
        {
            Извлечь из Q любую дугу (u,v);
            Если Revise(u,v), то
            {
                Добавить в Q все дуги вида (w,u).
            }
        }
```

Сложность алгоритма: $O(d^3 n^2)$.

Методы решения ЗУО

Распространение ограничений (например, достижение дуговой совместности с помощью AC-3) можно применять

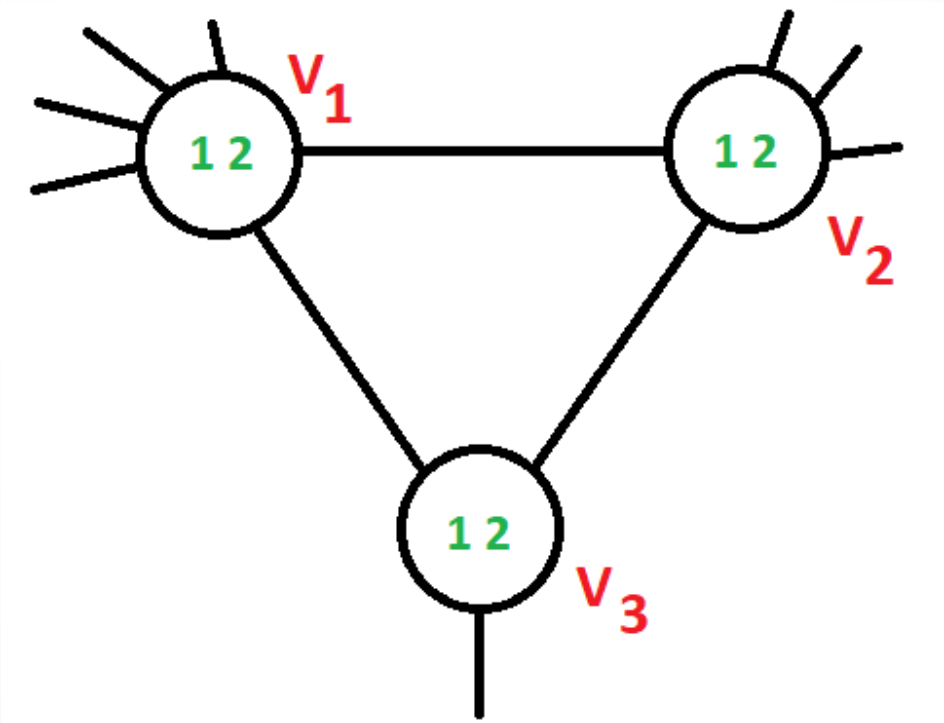
- Перед началом перебора (препроцессинг).
- В ходе перебора, после присваивания значений некоторым переменным.

После выполнения AC-3 текущий граф ограничений совместен по дугам. Возможны варианты:

1) Есть вершина (переменная) с пустой областью определения. $\Rightarrow$ ЗУО не имеет решения.
2) Для всех вершин область определения содержит 1 значение. $\Rightarrow$ ЗУО имеет единственное решение, и мы его уже нашли.
3) Иначе — ничего не ясно, надо выполнять перебор. Но за счёт уменьшения областей определения мы сократили время на перебор.

Методы решения ЗУО

3) Иначе — ничего не ясно, надо выполнять перебор.
Например: граф совместен по дугам, но не имеет решения.



Методы решения ЗУО

Можно обобщить определение совместности — ввести понятие k-совместности и строгой k-совместности. Приведя граф к строгой n-совместности, мы автоматически решим задачу. Но с увеличением k сложность алгоритма достижения k-совместности растёт экспоненциально.

Если граф ограничений — дерево, то достаточно привести его к дуговой совместности. После этого можно последовательно, начиная с «корня» дерева, присваивать переменным допустимые значения — конфликт не возникнет.

Методы решения ЗУО

Порядок перебора переменных:

- Эвристика MVR (Minimum Remaining Values). Из неприсвоенных переменных выбираем ту, у которой минимально количество оставшихся доступных значений. Эксперименты показали, что в практических задачах MVR даёт ускорение (по сравнению с «чистым» древовидным перебором) от 3 до 3000 раз.
- Степенная эвристика. Выбираем переменную, которая участвует в максимальном количестве ограничений с другими неприсвоенными переменными. Цель: быстрее сократить множества доступных значений для остальных переменных.
- (Идея, аналогичная степенной эвристике) Найти максимальное независимое подмножество вершин на графе ограничений (=переменных ЗУО) и это подмножество обрабатывать *в конце*. Поскольку варианты присваиваний этим переменным не зависят друг от друга, получим уменьшение сложности алгоритма: $O(d^{n-m} * md)$ вместо $O(d^n)$, где m — мощность такого подмножества.

Эвристика для определения порядка проверки значений текущей переменной: выбираем наименее ограничительное значение. Но эта эвристика требует бОльшего объёма дополнительных вычислений.